DISSERTATION

NETWORK DESIGN FOR RELIABILITY AND RESILIENCE TO ATTACK

by

Christian Klaus

March 2014

Dissertation Supervisor: Nedialko B. Dimitrov

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE 3–10–2014	**2. REPORT TYPE** Dissertation	**3. DATES COVERED** *(From — To)*

4. TITLE AND SUBTITLE NETWORK DESIGN FOR RELIABILITY AND RESILIENCE TO ATTACK	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Christian Klaus	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Navy	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense, or the U.S. Government. IRB Protocol Number: N.A.

14. ABSTRACT

We define and solve two network-design problems. In the first, (1) a defender uses limited resources to select a portfolio of paths or design a sub-network; (2) an attacker then uses limited attack resources to destroy network arcs, and then (3) the defender operates the damaged network optimally by finding a shortest path. The solution identifies a network design that minimizes post-attack path length. We show how the tri-level problem is equivalent to a single-level mixed integer program (MIP) with an exponential number of rows and columns, and solve that MIP using simultaneous row and column generation. Methods extend to network operations defined through general flow constructs. The second problem considers a stochastic logistics network where arcs are present randomly and independently. Shipping from a source to a destination may be delayed until a path connecting the two is available. In the presence of storage capacity, cargo can be shipped partway. The problem's solution identifies the storage locations that minimize the cargo's waiting time for shipment. We develop and demonstrate practical methods to solve this #P-complete problem on a model instance derived from a Department of Defense humanitarian shipping network.

15. SUBJECT TERMS

network design, network interdiction, network reliability, network resilience, stochastic network, warehouse allocation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	UU	149	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NETWORK DESIGN FOR RELIABILITY AND RESILIENCE TO ATTACK

Christian Klaus
Lieutenant Colonel, German Army
Diplom-Ingenier, University of the German Armed Forces Munich 2002
M.S., Operations Research, Naval Postgraduate School, 2011
M.S., Applied Mathematics, Naval Postgraduate School, 2011

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH
from the
NAVAL POSTGRADUATE SCHOOL
March 2014

Author: Christian Klaus

Approved by:

Nedialko B. Dimitrov Assistant Professor of Operations Research Dissertation Supervisor	Kevin Wood Distinguished Professor of Operations Research
W. Matthew Carlyle Professor of Operations Research	Michael P. Atkinson Assistant Professor of Operations Research
Timothy H. Chung Assistant Professor of Systems Engineering	

Approved by: Robert F. Dell
Chair, Department of Operations Research

Approved by: Douglas Moses
Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We define and solve two network-design problems. In the first, (1) a defender uses limited resources to select a portfolio of paths or design a sub-network; (2) an attacker then uses limited attack resources to destroy network arcs, and then (3) the defender operates the damaged network optimally by finding a shortest path. The solution identifies a network design that minimizes post-attack path length. We show how the tri-level problem is equivalent to a single-level mixed integer program (MIP) with an exponential number of rows and columns, and solve that MIP using simultaneous row and column generation. Methods extend to network operations defined through general flow constructs. The second problem considers a stochastic logistics network where arcs are present randomly and independently. Shipping from a source to a destination may be delayed until a path connecting the two is available. In the presence of storage capacity, cargo can be shipped partway. The problem's solution identifies the storage locations that minimize the cargo's waiting time for shipment. We develop and demonstrate practical methods to solve this #P-complete problem on a model instance derived from a Department of Defense humanitarian shipping network.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Algorithms

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ALB	Albania
AZE	Azerbaijan
B&B	Branch and Bound
BFS	breadth-first search
BGR	Bulgaria
BIH	Bosnia
CI	confidence interval
C-PPS-$\boldsymbol{m}$-$\boldsymbol{n}$	Continuous PPS-m-n where the attacker can continuously lengthen the arcs in the network up to a total length of n
C-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$	Continuous SNS-m-n where the attacker can continuously lengthen the arcs in the network up to a total length of n
DFS	depth-first search
DOD	Department of Defense
D-PPS-$\boldsymbol{m}$-$\boldsymbol{n}$	Delayed PPS-m-n where an attacked arcs is individually lengthened
D-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$	Delayed SNS-m-n where an attacked arcs is individually lengthened
ELA	equivalent link algorithm
FYR	Macedonia
GEO	Georgia
HRV	Croatia
LB	lower bound
LTU	Lithuania
LVA	Latvia
MDA	Moldova
MDP	Markov Decision Process
MDPD	Markov Decision Process with Design
MIP	Mixed Interger Linear Program
MNE	Montenegro
PPS-$\boldsymbol{m}$-$\boldsymbol{n}$	Path-Portfolio-Selection problem where the defender can select a set of m paths and the attacker can destroy n arcs in the network
Pr-PPS-$\boldsymbol{m}$-$\boldsymbol{n}$	Probabilistic PPS-m-n considering uncertainty about the value of n
Pr-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$	Probabilistic SNS-m-n considering uncertainty about the value of n
ROU	Romania
SNS-$\boldsymbol{m}$-$\boldsymbol{n}$	Sub-Network-Selection problem where the defender can select a sub-network containing no more than m arcs and the attacker can destroy n arcs in the network
SPNI	Shortest-Path Network-Interdiction problem
TSP	Traveling Salesman Problem
UB	upper bound
UKR	Ukraine
WLNR	Warehouse Location Network Reliability problem

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

This dissertation explores two network-design problems. The first problem, which has several variants, concerns the design of a transportation network that is resilient to worst-case attacks. The second problem concerns the design of a reliable logistics network, that can cope with random link failures.

Designing a Transportation Network That Is Resilient to Attack

The Path-Portfolio-Selection problem is a three-stage sequential game. In this game a defender selects a number of paths from a network, such that the shortest of those paths has minimum length under an optimal attack for a given attack budget. This dissertation begins with a detailed study on the Path-Portfolio-Selection-m-n problem (PPS-m-n), where the attacker destroys n arcs in the defender's portfolio of m paths from source to sink. An attacked arc in PPS-m-n is unusable for the defender. Under that assumption, we show that one can always construct an optimal portfolio in which only the longest path remains non-interdicted under the worst-case attack.

An example application for PPS-m-n is transportation of hazardous material, for which one wishes to have multiple shipping options in case of disruptions. The transportation requires time-consuming route preparation and clearance, which makes it almost impossible to deviate abruptly from the initial planning. A solution to an equivalent PPS-m-n model provides a set of paths that guarantee shipment under the worst-case disruption. A trivial solution is a set of independent, edge-disjoint, paths. The optimal solution, however, is not necessarily independent, i.e., paths can share a certain amount of edges, resulting in shorter post-attack responses than the trivial disjoint solution.

We formulate a single-level mixed integer linear program (MIP) that returns an optimal portfolio for PPS-m-n. The number of constraints for this MIP grows exponentially with size of the network because of the exponentially increasing number of possible attacks to consider. The MIP becomes impractical for large networks. We use a dynamic row-and-column generation approach that generates constraints and variables simultaneously to solve PPS-m-n for large networks. As the row-and-column generation procedure progresses, we solve smaller models that consider only subsets of all possible attacks against the network. Efficiency of solving such smaller models is improved by adding supervalid inequalities. Supervalid inequalities remove suboptimal candidate solutions from the solution space, and can thereby decease the solution time. Further savings in solution time are achieved by removing superfluous parts from the network.

Any feasible solution for the PPS-m-n model provides an upper bound for the maximum path length in the portfolio. Such an upper bound is used to remove unnecessary nodes and arcs from the network, that is, nodes and arcs that cannot appear in an optimal portfolio. The dissertation proposes a sequence of models, each model increasing in complexity, but also providing a tighter bound than the previous. Starting with the computationally cheapest model, we alternate solving a bounding model and shrinking the network based on the resulting bounding information. We develop a case study with a road network from Germany that consists of 2,971 nodes and 8,824 arcs, and solve PPS-5-2 in about two hours, after identifying as unnecessary more than 85% of nodes and arcs in the network. Here, the sequence of bounding models produces solutions within 1% of optimality in under seven minutes.

The dissertation defines and explores several variants of PPS-m-n. A probabilistic version, Pr-PPS-m-n, incorporates uncertainty in the attacker's capability. That uncertainty is expressed by a probability distribution on the number of attacked arcs. The solution to Pr-PPS-m-n is a portfolio with minimum expected length of the defender's response path. To compute the expected path length, one must identify the shortest non-interdicted path after an attack, requiring many additional constraints in the model. However, with a few modifications, the row-and-column generation approach and the network-simplification sequence apply to solve Pr-PPS-m-n to optimality. In addition to a computationally intensive method to find the optimal solution, we provide a simpler model that produces a near-optimal solution with a performance guarantee in a shorter amount of time.

An attacked arc in PPS-m-n or Pr-PPS-m-n is destroyed and unusable for the defender's post-attack operation. However, an attacked arc could still be usable, just less efficiently than if the arc is not attacked. Such inefficiency is modeled in D-PPS-m-n, where an attacked arc is lengthened rather than destroyed. Similar ideas and methods encountered while solving Pr-PPS-m-n guide us to formulate and solve D-PPS-m-n. The solution could be a portfolio of paths, all of which can be interdicted simultaneously. In the continuous relaxation of D-PPS-m-n, C-PPS-m-n, the attacker can "smear" the attack budget

continuously across the arcs in the network. The continuous relaxation of the attack variables enables a two-step reformulation of the inner optimization problem, leading to a single-level optimization model. The solution for C-PPS-m-n is a portfolio, containing disjoint paths with a minimum total length.

A closely related problem to PPS-m-n is the tri-level Sub-Network-Selection-m-n problem, SNS-m-n. The defender in SNS-m-n selects a sub-network containing no more than m arcs, such that the shortest path in the sub-network after the worst-case attack on n arcs has minimum length. Selecting a sub-network rather than a portfolio of paths gives the defender more flexibility, resulting in better responses than the one for an analogous instance of PPS-m-n. We provide models for SNS-m-n and its variations Pr-SNS-m-n, D-SNS-m-n, and C-SNS-m-n. Solving methods for PPS-m-n are adapted to solve SNS-m-n, returning optimal solutions that show similarities to the respective PPS-m-n optimal solutions. However, the SNS-m-n models can be extended to other models of network operations, such as minimum cost flow problems, which makes them particularly interesting.

Augmenting a Stochastic Logistics Network with Warehouses to Optimize Reliability

The second problem studied in this dissertations considers a stochastic logistics network, where arcs can fail randomly and independently. In this network, one wishes to ship cargo from a source to a destination. Because the network does not provide storage capacity in transshipment nodes, shipment must be delayed until the network provides a complete path from source to sink. In the presence of storage capacity (warehouses), cargo can be shipped partway, that is, from source to a warehouse and later to the sink. The Warehouse Location Network Reliability problem (WLNR) identifies the optimal storage locations that minimize the cargo's expected waiting time for shipment, and defines the associated optimal shipping policy. Actual transit time is negligible compared to the waiting time and is ignored.

WLNR is motivated by the current procedure used to ship humanitarian assistance cargo, using uncertain space available on recurring, scheduled military transportation routes. The uncertainty about space available renders the transportation network stochastic. The

Department of Defense (DOD) cannot store humanitarian assistance cargo at intermediate airports, unless a warehouse is created. The decision where to create and maintain warehouses is subject to budget constraints.

The solution for WLNR consists of two parts, the optimal location for the warehouses and the optimal shipping policy. The shipping policy specifies where to ship the cargo, in particular, it ships to the reachable node with storage capacity, a warehouse or sink, from which one expects the smallest shipping time to the sink. We demonstrate a connection between WLNR and a Markov Decision Process with Design (MDPD), which provides both the optimal warehouse location and the optimal shipping policy. For large networks, a MDPD is intractable, because of a large state space. However, the results from the MDPD reveals the structure for the optimal shipping policy. For fixed warehouse locations, the MDPD with pre-defined shipping policy behaves like a Markov chain. We use such a Markov chain to compute the expected shipping time for one or two warehouses. The resulting equations involve several s-t-reliabilities. The s-t-reliability is the probability that a path exists between two nodes, s and t, in a stochastic network. Computing s-t-reliability is known to be #P-complete, and impracticable for large networks.

We provide methods to find the optimal warehouse locations for one and two warehouses. Estimating reliability by Monte Carlo simulation, which is very time-consuming, lies at the core of the methods to solve WLNR. We show theoretical results that allow the elimination of suboptimal solutions from the solution space, resulting in a significant saving of computational effort. We create a case study from the DOD's humanitarian assistance transportation network with 95 nodes and 1096 arcs. There are more than 8000 possible ways to place two warehouses in that network. We identify up to 95% of these combinations as being suboptimal without computing the associated shipping times, which saves on simulation effort. We achieve further savings of simulation time by customizing the sampling method. A typical procedure to estimate s-t-reliability draws a sample of the stochastic network, and then verifies whether s and t are connected in that sample. We use a technique that only samples arcs as needed while searching for a path from s to t. In the case study, we sample less than 20% of all arcs when applying this technique. We also provide a theoretical bound on the best possible expected waiting time. The bound shows on the case study example that adding more than two warehouses will improve results only fractionally.

Acknowledgments

Before I try and probably fail to express my gratitude to all the people that have played a part in getting me to where I am today, I will constrain myself to a short list. To those whose names do not appear, I offer an apology and want you to know that I am thankful for any help I received.

First and foremost, I would like to thank my advisor, Professor Nedialko Dimitrov, for his mentorship and friendship over the last few years. From the first day we met, you set the bar high and believed we could make it. Now you proved yourself right, and we all know how much you love proving things. It was truly a pleasure to work with you, and I will miss our weekly discussion meetings.

Second, I would like to thank my committee members, professors Mike Atkinson, Matt Carlyle, Tim Chung and Kevin Wood, for their support and advice. Stepping up and joining the dissertation committee added just another load to your already busy schedules. I'll never forget it.

Third, to my office-mates, Matt and Gary, I say thanks for friendship and all the little, but always welcome disruptions in our humble kingdom. It was a blast to get to know you guys, and I wish you the best of luck for the future. Special thanks go to Jesse, who in my opinion, finished the program too early. I enjoyed our discussions and missed them after you left.

Last, but certainly not least, I would like to thank my loving wife, Ilka, and our outstanding sons, Julius and Eric. Certainly, I would not be where I am today if it were not for your love and support. I am truly grateful and blessed by you each and every day.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

This dissertation concerns two network design problems. In the first problem, we design a network that can cope with worst-case, deliberate failures. In the second problem, we design a network that can cope with random link failures.

Modern life depends on many interdependent network infrastructures. Electrical power networks provide energy to keep our homes running. Logistics networks rely on road and railway networks to meet demand with supply. We build social networks with friends, and stay in touch via networking services such as Facebook. The social network services themselves rely on telecommunication networks and the Internet. Networks are an integral part of our everyday life.

Each network serves a purpose, and how well the network achieves that purpose can be expressed with a network performance measure (e.g., maximum flow, or shortest distance from a source to a destination). To achieve the highest possible network performance, one needs to operate the network appropriately. Such optimal network operations are the solutions to variants of network optimization problems, such as, shortest path problems, maximum flow problems, and minimum cost flow problems. Typically, a network's performance is restricted by the weakest part in the network. Identifying and improving weak parts of a network leads to network designs with higher performance. Designing such networks requires solving different optimization problems.

Mathematically speaking, a network is a collection of nodes that are connected by arcs. Nodes and arcs in a network can fail and thereby affect the network's performance. For example, an electrical power grid can be modeled as a network, where power plants are represented by nodes. Such a node could fail, causing a chain reaction and blackout. An example for a network with arc failures is a road network in which a particular bridge is modeled as an arc. Removing that arc from the network is equivalent to closing that bridge, which could affect hundreds of commuters and cause traffic jams at other locations.

According to the source of a failure, we distinguish two basic types: deliberate and unintentional failures. An unintentional failure is a random failure that just happens, e.g., a power plant goes down because of age. A deliberate failure is induced intentionally by an adversary to cause disruption and/or gain advantage. For instance, a bridge in a road network could be a strategically high value military target, and its destruction could give the most disadvantage to a defender, e.g., it cuts all supply chains. To gain advantage, an attacker will try to attack that bridge. We define *network reliability* to mean a network's ability to perform well under random failures. We say a network is *resilient to attacks* if it performs well under worst-case deliberate failures.

1.1 Resilience of a Network to Attack

Non-technically speaking, and according to Webster's dictionary, resilience is the ability to recover from or adjust easily to a misfortune or change. Besides the non-technical meaning in American language, resilience is associated with subject-specific technical meanings. The word resilience is used in the field of critical infrastructure, independence and robustness [1]. Chen and Miller-Hooks define resilience as an indicator of recovery capability from a disaster in intermodal freight transport [2]. They formulate a stochastic mixed integer program, and a solving procedure that combines decomposition techniques and Monte Carlo simulation. Colbourn defines resilience as the expected number of node pairs that are connected in a stochastic network, and is focused of computing that number [3]. The first part of this dissertation builds on the idea of selecting a set of options that is resilient to attacks. Menth and Martin consider a related idea in their work [4]. They propose a multi-topology routing in a computer network, which makes it resilient against random link failures. All this previous work is focused on assessing reliability, assuming random failures. This dissertation concerns designing networks that are robust to worst-case failures, which is a conservative approach.

In the field of Operations Research, we typically identify worst-case failures with network-interdiction problems [5, 6, 7]. As an example of a network-interdiction problem, an attacker could destroy n arcs of the network, and the defender could use the remaining shortest path from source to sink in the network for operation. The network's resilience to attacks can be visually described by an operator resilience curve, which plots the length of the shortest remaining path on the vertical axis versus the number of attacks on the horizontal axis. The same visual description is applied to other network problems, e.g., maximum flow problems [8]. Network-interdiction problems are, in general, bi-level problems. In the first level, the attacker selects an attack; in the second level, the defender uses the remaining network for operation, e.g., shipping across the shortest path, pushing as much flow as possible through the network, or satisfying supply and demand in the most cost-efficient way. This dissertation concerns tri-level optimization problems. A tri-level optimization problem builds on the bi-level problem by allowing the defender an additional first move to protect parts of his operations. Defender's and attacker's opposing goals result in tri-level optimization problems, where the objective function value in each level is driven in the opposite direction than in the previous level. Brown et al. define a defender-attacker-defender model to make critical infrastructure more resilient against terrorist attacks, which is tri-level [9].

This dissertation considers a number of tri-level network optimization problems. One such problem is the Path-Portfolio-Selection problem. In the Path-Portfolio-Selection-m-n (PPS-m-n) problem, a defender first selects a portfolio of m paths from an origin to a destination; second, an attacker destroys n arcs from the selected paths; third, the defender uses the shortest remaining path in the portfolio for operations. PPS-m-n can be viewed as designing a network that is resilient to attacks, where the design occurs through the selection of the portfolio. Selecting a portfolio, in which the paths maintain a certain level of disjointness is key to increase the resilience. Our models do not require a portfolio of independent, edge-disjoint paths, but as the number of attacks approaches the number of paths in the portfolio, the resulting optimal solution eventually becomes independent.

Transportation of hazardous material, e.g., nuclear waste to a disposal site, motivates PPS-m-n. Shipping hazardous material requires planning a long time in advance. Size, shape and weight of the transportation containers are limiting factors; some areas cannot be passed by hazardous material transports, e.g., watersheds, and some parts of the shipping route may require special preparation or permission for use. Once the preparation for shipment is done, deviating from the initial planning is impossible. In Germany, for instance, transportation of nuclear waste to disposal sites is often disrupted by protests and demonstrations [10]. Nuclear opponents block roads and train tracks, and thereby increase transit time and associated vulnerability of the transports. The solution to PPS-m-n can provide a shipping plan that is resilient against a certain number of disruptions. Other applications are transportation of very important persons, for which one wishes to have multiple options that are resilient against disruptions, caused by protesters or attackers.

A second, related, network resilience problem is the Sub-Network-Selection-m-n problem (SNS-m-n). SNS-m-n is also a tri-level problem where we design a network that is resilient to attacks. The only difference to PPS-m-n is that in the first level, the defender selects a sub-network consisting of m edges as opposed to selecting m paths. Then, after the attack, the defender is allowed to use any arc in the selected sub-network. SNS-m-n can be thought of as selecting the most resilient sub-network of size m from the original network.

The dissertation also addresses several variants of PPS-m-n and SNS-m-n, both in terms of creating models and practicable solution approaches. Our solution methods critically depend on constraint and variable generation in formulations that are exponential in both. To improve solution times, we introduce several supervalid inequalities that remove suboptimal solutions from consideration, and also solve a sequence of problems to reduce network size. While this dissertation is focused on shortest path network operations, our results could be generalized to arbitrary minimum cost flow network operations.

1.2 Reliability of a Logistics Network with Uncertain Links

A well studied problem in the literature addressing network reliability is the s-t-reliability problem [11]. In the s-t-reliability problem, arcs of a stochastic network randomly and independently fail, and one seeks to measure the probability that two nodes, s and t, are connected. Even measuring this probability is a difficult, #P-complete problem [12]. In the literature, there have been studies of many variants to calculate, bound or estimate the s-t-reliability [13, 14, 15].

This dissertation considers improving a logistics network's poor reliability. Specifically, we consider a variant of the s-t-reliability problem, the Warehouse Location Network Reliability (WLNR) problem. In WLNR we consider a stochastic network and fixed time periods. In every time period, a link exists randomly according to a probability distribution. An item is to be shipped across that stochastic network, but has to wait for shipment until the network contains a complete path from s to t. The reciprocal of the s-t-reliability is the expected number of time periods until such a path exists. For simplicity, we call that number expected shipping time. Introducing warehouses, storage capacity, to some of the nodes, allows partway shipments, i.e., shipping from s to a warehouse and later to t, which can decrease the shipping time. The WLNR problem finds the warehouse locations which minimizes the expected shipping time.

WLNR is motivated by an application that is shipping humanitarian material across DOD logistics transportation routes. Humanitarian material can be shipped only if unused space is available on a recurring scheduled route, but the availability of such space is uncertain. Space availability can be modeled as random failure of an arc; if space is available, the arc is present; otherwise the arc is not present. The Department of Defense cannot store humanitarian material at intermediate airports unless a warehouse is created and paid for in advance. The WLNR problem identifies the transshipment nodes, airports, where it would be best to install such a warehouse.

The remainder of the dissertation is structured as follows. In Chapter 2, we define and study the network resilience problems PPS-m-n, SNS-m-n and their variants. In Chapter 3, we define and study the WLNR problem. Each of the chapters start with a more detailed introduction and background discussion. We add concluding remarks and future directions of study in Chapter 4.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Designing a Transportation Network That Is Resilient to Attack

2.1 Abstract

This chapter defines the Path-Portfolio-Selection-m-n problem, PPS-m-n. The problem concerns a three-stage sequential game model where (1) a defender selects a portfolio of m paths from a source to a sink node in a directed network, (2) an attacker removes n arcs from the network, and (3) among all the paths in the portfolio from which no arc has been removed, the defender selects the shortest for operation. The optimal solution for PPS-m-n is a portfolio, such that the path chosen in response to a worst-case attack has minimum length. Even the simplest such problem is NP-complete, since PPS-2-1 is equivalent of finding two arc-disjoint paths in a network, such that the longest of these paths is of minimum length. We show that PPS-m-n is equivalent to a single-level mixed integer program, in which the number of constraints grows exponentially with network size. We create practicable solving methods for large instances using a row-and-column generation approach. We propose a sequence of bounding models, each model increasing in complexity, but also producing a better bound on the objective function value than the last. The bounds are used to sequentially remove superfluous parts of the network. Two case studies with large road networks (2,500 or 45,000 nodes) show that the sequential bounding method can quickly yield solutions within 1% of optimality.

This chapter also defines and explores a number of variants of PPS-m-n. In a probabilistic version Pr-PPS-m-n, we incorporate uncertainty about the attacker's budget, which is expressed by a probability mass function for the number of arcs attacked. We also define and provide solution methods for (1) the "delayed" D-PPS-m-n, where an attacked arc is lengthened rather than destroyed, and (2) the continuous relaxation of that model denoted, C-PPS-m-n with an attack budget continuously spread across the arcs in the network. A closely related problem to PPS-m-n is the Sub-Network-Selection-m-n problem, SNS-m-n. The defender in SNS-m-n selects a sub-network containing no more than m arcs, such that the shortest path in the sub-network after the worst-case attack on n

arcs has minimum length. We discuss models and provide solving methods for SNS-m-n and its probabilistic, delayed, and continuous versions. These models can be extended to other models of network operations, such as minimum cost flow.

2.2 Introduction

Transportation of hazardous materials can pose a danger to the public and exposes unsafe material to disasters and attack. To limit the risk involved, one wishes to keep the transit time as low as possible. In Germany, for instance, transportation of nuclear waste to disposal sites is often accompanied by protests and demonstrations [10]. Nuclear opponents block roads and train tracks, and thereby increase transit time, which keeps the cargo vulnerable longer than necessary. When potentially delay-causing activities are observed, a transportation operator would like to reroute the transport, but abrupt route changes are not always possible. Size, shape and weight of the transportation containers are limiting factors; some areas cannot be passed by hazardous material transports, e.g., watersheds, and some parts of the shipping route may require special preparation or permission for use. Typically, route planning and preparation is time-consuming and done a long time in advance.

We model a situation in which the operator has a set of preplanned shipping routes, a portfolio of options, from which he must choose one for operation. According to the observed behavior of nuclear opponents, he selects the route with smallest delay. The portfolio adds some flexibility of response and the operator seeks a portfolio that provides the minimum delay under the worst-case disruption. The restriction of executing an option from a preselected set of options has many examples in real life. In the nuclear waste transportation example, it may require a lot of time and effort to prepare a route for transportation, or to get permission to pass through an area. Another example is transporting very important persons. To keep a certain level of safety, different transportation routes are often prepared long in advance, and abrupt, last minute changes are impossible.

The example described has the features of a defender-attacker-defender model [9]. The operator in the example acts as defender and the nuclear opponents assume the role of the attacker. The defender wants to ship cargo from a *source* to a *sink* across a sequence of road segments, a shipping *path*. The attacker tries to disrupt the defender's shipment. Defender and attacker play a sequential game modeled in three stages. (1) The defender selects a flexible portfolio of m paths, which offer both short paths and robust options in case of an attack. (2) The attacker is aware of the defender's planning. He knows about the paths in the portfolio and appropriately attacks n arcs in the transportation network. The attack plan is designed to force the defender's shipment on a path that is as long as possible. (3) The defender observes the attack, must resort to an option from his portfolio, and selects the remaining shortest path. Defender and attacker have the same objective functions, the length of the final shipping route, but the defender tries to minimize that length, while the attacker tries to maximize it. This is a zero-sum situation, where gains for one opponent are losses for the other. We call this three-stage sequential game the *Path-Portfolio-Selection-m-n* problem, PPS-m-n.

Closely related to PPS-m-n is the *Sub-Network-Selection-m-n* problem, SNS-m-n. Similar to PPS-m-n, SNS-m-n is a three-stage game. In the first stage, given a source and a sink, the defender selects a sub-network consisting of no more than m arcs from the original network. In the second stage, the attacker attacks n arcs of the sub-network. In the third stage, the defender routes cargo from the source to the sink in the attacked sub-network. The main difference between PPS-m-n and SNS-m-n is the flexibility the defender has to select a route after the attack. In PPS-m-n, the defender must stick to one of the routes in the portfolio. In SNS-m-n, the defender can select any route in the attacked sub-network.

Fortifying infrastructure against natural disaster is an illustrative example for SNS-m-n. Consider a natural disaster, a hurricane for example, that destroys all of the network infrastructure in an area, modeled as arcs. Ahead of time, one can harden at most m arcs against the disaster, but even some of the hardened arcs, namely n, are destroyed. The SNS-m-n model can be used to identify the sub-network to harden, such that a network operation on the hardened infrastructure after the worst-case failure of n components performs in the best possible way. This dissertation primarily discusses shortest path network operations; however, the problem definition and some of the results can be extended to arbitrary network optimization problems, such as network flow problems.

2.2.1 Background and Related Work

Routing of hazardous material shipments initially received attention in the literature to balance the tradeoffs between cost minimization and outcomes that would stem from an accident. Bell [16] investigates repeated shipments of hazardous materials with unknown probabilities of arc incidents, and shows that the safest strategy is a mixed route strategy, that is, selecting routes randomly according to a predefined probability. Later work, similar to the example sketched in the introduction, incorporates the threat of intentional disruptions, e.g., terrorist attacks [17], leading to bi-level attacker-defender optimization problems. Katurska et al. [18] introduces a third decision stage and investigates how infrastructure protection can impact the optimal solution. He illustrates the benefit of mixed route strategies by transporting a very important person.

All three references have similarities to the introductory example, but in contrast to PPS-m-n, they all use mathematical models based on probabilities of using specific routes and probabilities of attacking specific arcs in the network. The probabilities can be treated as data or as decision variables, leading to different solution approaches like stochastic programs or simultaneous game theoretic solving approaches involving equilibria and pay-off matrices. Although PPS-m-n and SNS-m-n are three-stage games, they are sequential, involve exponentially sized strategy spaces, and their solutions are significantly different from the previous game theoretic work.

The transportation operator from the introductory example wishes to minimize delays and ship across the shortest possible path. Identifying the shortest path in a network is a common network optimization problem and is widely used in applications. There is a variety of methods available to solve such problems, ranging from Linear Program formulations to more efficient algorithms [19]. Here, the operator can preselect m paths. The problem of finding the m shortest paths in a network is a more difficult optimization problem, and is known in literature as k shortest loopless paths [20, 21]. A closely related problem to the k shortest path problem is the near shortest path problem, which is to identify all paths in a network that are no longer than the shortest path plus an additional length ϵ. Algorithms with different space and time complexity are available to solve a k shortest path problem or a near shortest path problem [22].

A special set of paths is a set of pairwise disjoint paths, that is, no arc in the network can be part of any two or more paths. Special instances of PPS-m-n always have a portfolio of disjoint paths as solution. Finding two disjoint paths such that the length of the longer path is minimized is proven to be strongly NP-complete [23].

The core of PPS-m-n and SNS-m-n is a network-interdiction problem. Network-interdiction problems involve two opponents, a defender and an attacker, with opposing objectives. Such models are often called attacker-defender models. Relevant for our research is the *Shortest-Path Network-Interdiction* problem, SPNI. Fulkerson and Harding [5] maximize the shortest path within a given attack budget, while Golden [6] enlarges the shortest path by at least τ units, using a least cost investment strategy. Fulkerson, Harding and Golden assume the attack variables to be continuous, and continuously distribute the attack budget among the interdicted arcs. That assumption enables them to formulate linear programs to solve their problems. The variations C-PPS-m-n and C-SNS-m-n follow the same ideas and allow continuous attack variables. However, PPS-m-n and SNS-m-n are discrete in the sense that the attacker selects a small set of arcs to interdict. Shortest path network-interdiction problems with binary attack variables can still be solved by standard linear-programming based solvers [19] using branch-and-bound techniques, but more efficient algorithms exist, e.g., the algorithms developed by Wood and Israeli [7] based on Benders' decomposition [24].

The network-interdiction problems are bi-level, but PPS-m-n and SNS-m-n are tri-level optimization problems; in particular, tri-level problems where the inner level drives the objective in the opposite direction than the outer levels of the problem. An example for such a tri-level optimization model is a defender-attacker-defender model [9]. Such a defender-attacker-defender model is built on the traditional attacker-defender model by allowing the defender to first protect some parts of his operation, before an attack occurs. The resulting solutions can give recommendations on how to improve the system's resilience.

2.2.2 Problem Statement and Organization

We consider a directed network $G = (N, A)$, with nodes $n \in N$ and arcs $(i, j) \in A$, where each arc has an associated nominal arc length c_{ij}. Source s and sink t are distinguished nodes. A defender wants to ship material from source to sink and the shipping cost is proportional to the length of the shipping path. A shipping path is an ordered sequence of arcs connecting s and t.

In PPS-m-n, the defender can select a portfolio P of m paths from which the attacker can interdict at most n arcs. An interdicted arc is destroyed and unusable for the defender. An *attack plan* is a set of n arcs, each interdicted if the attack plan is executed. In contrast to a shortest path network-interdiction problems, the defender is restricted to responding with a path out of P. He chooses the shortest of such paths from which no arc has been destroyed. The PPS-m-n problem is to select a portfolio of m paths, such that the path chosen in response to a worst-case attack of n arcs has minimum length.

In SNS-m-n the defender can extract a sub-network of at most m arcs, from which the attacker can destroy at most n arcs. The defender responds to the attack with the remaining shortest path in the attacked sub-network. The SNS-m-n problem is to select a sub-network of m paths, such that the remaining shortest path after the worst-case attack on n arcs has minimum length.

PPS-m-n as well as SNS-m-n assume that an attacked arc is destroyed. In the real world, this may not be the case. An attacked arc could still be usable for the defender but lengthened by a predefined length. Now, the defender could be better off using an interdicted arc and accepting a delay that is caused by the additional arc length. We define D-PPS-m-n and D-SNS-m-n, where the attribute "D" stands for *delay* and indicates that an attacked arc is lengthened by the fixed value d_{ij}. We also define continuous version of the delayed problems, C-PPS-m-n and C-SNS-m-n, that do not consider the parameters d_{ij}, but the attacker can continuously distribute the attack budget n across the arcs in the network, and thereby, individually lengthen the attacked arcs. The attack budget n represents a total length that can be added to the network.

PPS-m-n and SNS-m-n use the parameter n, representing the attack budget. Setting a specific value for n requires perfect knowledge about the attacker's capability. Typically, we do not have perfect knowledge about the attacker's available resource. The probabilistic versions, Pr-PPS-m-n and Pr-SNS-m-n, incorporate attack probabilities to account for uncertainty about the attacker. We define attack probabilities as the probability that $\ell = 1, \ldots, n$ arcs are attacked by the attacker. The solutions for Pr-PPS-m-n and Pr-SNS-m-n yield an expected response length that is potentially shorter than the expected response length resulting from the PPS-m-n and SNS-m-n solutions because there is some chance that the attacker uses less than n attacks.

In Section 2.3 we develop a basic MIP based on a shortest path and multi-commodity flow formulation to solve the tri-level PPS-m-n optimization problem. We enhance that basic linear program with a number of additional constraints, supervalid inequalities, that can help reduce solution time. Nevertheless, the number of constraints in the formulation quickly grows with network size. We show how PPS-m-n can be solved for large sized networks using a row-and-column generation approach. In addition, we explore two special cases of the problem, namely PPS-m-1 and PPS-m-$(m-1)$ that can be solved directly with a single optimization model. Section 2.3.5 gives a sequence of steps that can remove superfluous parts of the network, which further reduces the solution time for PPS-m-n. Section 2.3.6 explores Pr-PPS-m-n. We give a general MIP that incorporates attack probabilities, and a bounding model that does not guarantee to return the optimal solution to Pr-PPS-m-n but a solution with a performance guarantee in a shorter amount of time. We also explore the special cases Pr-PPS-m-1 and Pr-PPS-m-$(m-1)$. In Section 2.3.7 we show a model that theoretically returns the solution to D-PPS-m-n, and its continuous equivalent, C-PPS-m-n, is examined in Section 2.3.8. Solution methods and solution structures for SNS-m-n, D-SNS-m-n, Pr-SNS-m-n and C-PPS-m-n have similarities to the equivalent PPS-m-n problems, and are examined in Section 2.4. Section 2.5 summarizes the examples and computational results encountered throughout the paper.

2.3 The PPS-m-n Problem

This section mathematically defines PPS-m-n. We develop a MIP to solve PPS-m-n and present supervalid inequalities that can help to reduce solution time. The model has exponentially many constraints in the size of the network, and we propose a row-and-column generation approach to solve PPS-m-n on large networks, where the full MIP cannot be efficiently written from the start.

Each of the following sections builds on the previous, starting with some additional notation that may be required for the section. We begin with some general notation, and a simple but intractable formulation of the problem. For a given directed graph $G = (N, A)$ PPS-m-n is unambiguously defined by the two integers m and n.

Sets and indices

$i, j \in N$	nodes in network G; s and t are distinguished as source and sink
$(i, j) \in A$	arcs in G
$E \in \mathcal{E}$	$\mathcal{E}$ is the set of all cardinality n subsets of A, representing all possible attack plans; the attacker interdicts all arcs in E for a specific attack plan $E \in \mathcal{E}$
$\bar{y} \in \overline{Y}$	$\overline{Y}$ is the set of all simple paths from s to t in G; $\bar{y} \in \{0,1\}^{\lvert A \rvert}$, where $y_{ij} = 1$ if the arc (i, j) is in the path, and $\bar{y}_{ij} = 0$ otherwise; also we let $l(\bar{y})$ be the associated path length
$P \in \mathcal{P}$	$\mathcal{P}$ is the set of all possible portfolios, each containing m s-t-paths
$k \in K$	k is an index, pointing to the k^{th} path in the defender's portfolio; the portfolio has m paths and $K = \{1, \ldots, m\}$

Data

c_{ij}	nominal length of the arc (i, j)
$l(k)$	length of the k^{th} path in the defender's portfolio
m	defender's resource; number of paths in the defender's portfolio
n	attacker's resource; number of arcs that can be attacked simultaneously

PPS-m-n is a three-stage problem. In the first stage, the defender selects a portfolio of m paths. In the second stage, the attacker attacks n arcs. In the third stage, the defender selects the shortest path remaining in his portfolio. Let $l(p, E)$ be the length of path p under attack plan E. The following is a concise formulation of PPS-m-n:

$$\min_{P \in \mathcal{P}} \max_{E \in \mathcal{E}} \min_{p \in P} \; l(p, E). \tag{2.1}$$

Difficulties with standard approaches. Formulation (2.1) illustrates the tri-level optimization of PPS-m-n. However it does not lead to a practical algorithm for solving the problem. The most closely related work, that on network interdiction, typically concerns bi-level problems. For bi-level problems, a common algorithmic technique is Benders' decomposition [24]. If the second level of the bi-level optimization problem involves continuous variables, then reformulating the second level optimization problem in terms of dual variables will yield a single-level optimization problem [7]. The optimal solution for such a single-level optimization problem equals the optimal solution for the original bi-level optimization problem. These approaches do not work here because both the first and second level of optimization are integer-constrained. To illustrate the difficulties with solving a tri-level optimization problem, we follow through with a Benders' decomposition approach, resulting in (2.2).

Additional notation

d_{ij}	additional length of interdicted arc (i, j)
M	a large positive value, bigger than a minimum we specify later to serve the model purpose
$w_{\bar{y}}$	1 if path $\bar{y}$ is chosen to be in the defender's portfolio, and 0 otherwise
x_{ij}	1 if arc (i, j) is interdicted, and 0 otherwise
$\underline{z}$	length of the shortest path remaining in P after an attack

Unsophisticated formulation for PPS-*m*-*n*

$$\min_{\boldsymbol{w}} \max_{\underline{z},\, \boldsymbol{x}} \ \underline{z} \tag{2.2a}$$

$$\text{s.t.} \quad \underline{z} \le l(\bar{y}) + \sum_{(i,j)|\bar{y}_{ij}=1} d_{ij}x_{ij} + M(1 - w_{\bar{y}}) \qquad \forall \bar{y} \in \overline{Y} \tag{2.2b}$$

$$\sum_{\bar{y}\in\overline{Y}} w_{\bar{y}} \le m \tag{2.2c}$$

$$\sum_{(i,j)\in A} x_{ij} \le n \tag{2.2d}$$

$$w_{\bar{y}} \in \{0, 1\} \qquad \forall \bar{y} \in \overline{Y} \tag{2.2e}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{2.2f}$$

Model (2.2) has the structure of Benders' decomposition on a shortest-path network-interdiction problem [7], except the shortest path is selected out of the available portfolio. The $w_{\bar{y}}$ variables turn off all constraints for paths that do not belong to the portfolio. The value for M renders constraint (2.2b) vacuous for all not selected paths. While setting M to a large positive value serves the modeling purpose, it can raise computational issues and long solution times if not handled with care [25]. Here, a value greater than the length of the longest simple path in the network is sufficiently large. Constraints (2.2c) and (2.2d) are the resource constraints for the defender and attacker, respectively. The optimal attack plan corresponding to the chosen portfolio is retrieved from the $\boldsymbol{x}$ variables.

Model (2.2) is not solvable for several reasons. First, some variables in the model are maximizing while others are minimizing, meaning standard optimization algorithms are not directly applicable. A typical approach to fix this is to take the dual of the inner optimization problem; however, the inner problem in the formulation is an integer program. Taking the dual of the inner problem does not work because of the integer duality gap that would be introduced. Second, the formulation above has too many constraints, one for each path from source to sink. It is impossible to list all those efficiently, although there is potential to use a clever row-generation technique. Model (2.2) highlights the fact that standard approaches to solving the PPS-m-n problem, even those that borrow from interdiction models, are intractable.

2.3.1 An Effective Formulation for PPS-m-n

For now, we concentrate on destroying arcs, as defined for PPS-m-n, because it is simpler to reason about this case. Sections 2.3.7 and 2.3.8 discuss variants on how to relax this, so that attacking an arc simply lengthens it. The following model, even though it has an exponentially large size, is a single-level MIP for solving PPS-m-n.

Additional notation

y_{ij}^k	1 if arc (i,j) is in path k in P, and 0 otherwise
$\overline{z}$	length of the longest path in P
ι_E^k	1 if attack plan E interdicts path k, and 0 otherwise; in other words, $\iota_E^k = 1$ if at least one $(i,j) \in E$ is in path k

Single-level model for PPS-$\boldsymbol{m}$-$\boldsymbol{n}$

$$\overline{z}^* = \min_{\boldsymbol{y},\,\boldsymbol{\iota},\,\overline{z}} \quad \overline{z} \tag{2.3a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij}^k - \sum_{i:(j,i)\in A} y_{ji}^k = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall k \in K \tag{2.3b}$$

$$\overline{z} \geq \sum_{(i,j)\in A} c_{ij} y_{ij}^k \qquad \forall k \in K \tag{2.3c}$$

$$\iota_E^k \geq y_{ij}^k \qquad \forall k \in K,\ (i,j) \in E,\ E \in \mathcal{E} \tag{2.3d}$$

$$\iota_E^k \leq \sum_{(i,j)\in E} y_{ij}^k \qquad \forall k \in K,\ E \in \mathcal{E} \tag{2.3e}$$

$$\iota_E^k \leq 1 \qquad \forall k \in K,\ E \in \mathcal{E} \tag{2.3f}$$

$$\sum_{k\in K} \iota_E^k \leq m - 1 \qquad \forall E \in \mathcal{E} \tag{2.3g}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ (i,j) \in A \tag{2.3h}$$

In model (2.3), we assign each path in P a different commodity, distinguished by commodity numbers, k. Each commodity travels along its assigned path from s to t. Constraint (2.3b) is the flow conservation constraint for each commodity, selecting a path from s to t. The variable ι_E^k is an indicator whether path k is interdicted by attack plan E. Constraints (2.3d), (2.3e), and (2.3f) ensure that ι_E^k is one if attack plan E interdicts path k and zero otherwise. Constraint (2.3g) ensures that no attack plan of cardinality n can interdict all the paths in the portfolio simultaneously. That leaves at least one non-interdicted path as the defender's response. We show shortly, that under an optimal attack, the defender's response is always the longest path in the portfolio. Hence, the objective function is the length of the longest path in P by (2.3c). We use binary variables to select paths in the portfolio with constraint (2.3h). The indicator variables ι_E^k are binary by definition, but their integrality is relaxed in the model. However, they take on integer values automatically by (2.3d)–(2.3f). Model (2.3) is only feasible if $m > n$, which we assume for all models, where an attacked arc is destroyed. It remains to be shown that model (2.3) returns the optimal solution for PPS-m-n. We begin with a useful building block, discussing a model for computing optimal attacks on a given portfolio.

Attacking a given portfolio

Given a portfolio of paths, indexed by the set K, the attacker seeks to interdict a set of n arcs in the network, such that the remaining shortest path in the portfolio is of maximum length. Model (2.4) computes the attacker's optimal attack. From the attacker's point of view, the paths in the portfolio are data, reflected by the notation below.

Additional notation

$\hat{y}_{ij}^k$	1 if arc (i,j) is in the k^{th} path of the given portfolio, and 0 otherwise

Attacker's optimization model for a given portfolio

$$\underline{z}^* = \max_{\boldsymbol{x},\, \underline{z}} \ \underline{z} \tag{2.4a}$$

$$\text{s.t.} \quad \underline{z} \leq \sum_{(i,j)\in A} \hat{y}_{ij}^k (c_{ij} + M x_{ij}) \qquad \forall k \in K \tag{2.4b}$$

$$\sum_{(i,j)\in A} x_{ij} \leq n \tag{2.4c}$$

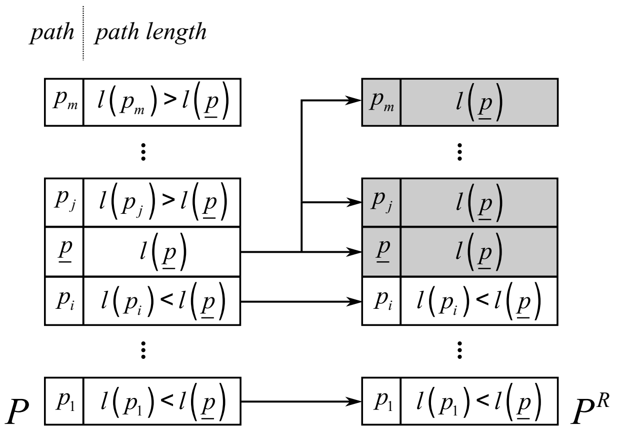
$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.4d}$$

The right hand side of constraint (2.4b) expresses the length of a path under attack plan $\boldsymbol{x}$. Here, big-M represents infinity, because an attacked arc is destroyed. A value of $M = |N| \cdot c_{ij}^{\max}$, where $c_{ij}^{\max}$ being the length of the longest arc in the network, is sufficiently large, and serves the model purpose. The attacker can increase the length of at most n arcs, enforced by the budget constraint (2.4c). The defender selects the shortest path remaining in the portfolio as response to the attack, expressed by the objective function (2.4a). This formulation is similar to standard Benders' decomposition for shortest-path network interdiction [7].

Proof that model (2.3) solves PPS-m-n

The formulation (2.3) relies on the assumption that the defender's response to the optimal attack is always the longest path in the optimal portfolio. That means, all the paths but the longest paths in the optimal portfolio are interdicted by the optimal attack plan E^*. This section proves that such an optimal portfolio can be constructed, and that model (2.3) returns an optimal portfolio as least as good as the constructed one.

We define an *operational* portfolio of paths to be a portfolio where the optimal attack leaves at least one non-interdicted path. Constraints (2.3d)–(2.3g) ensure that any feasible portfolio is operational as well. Furthermore, we define a procedure $P^R = \textsc{Replace}(P)$, that takes as input an operational portfolio P and produces as output a new portfolio P^R. The procedure is described pictorially in Figure 2.1. In words, assume that the optimal attack on portfolio P does not interdict several paths, but the shortest non-interdicted path is $\underline{p}$. The REPLACE procedure replaces all paths in P that are longer than $\underline{p}$ with a copy of $\underline{p}$.



path
path length
p_m
$l(p_m) > l(\underline{p})$
p_j
$l(p_j) > l(\underline{p})$
$\underline{p}$
$l(\underline{p})$
p_i
$l(p_i) < l(\underline{p})$
p_1
$l(p_1) < l(\underline{p})$
P
P^R

Figure 2.1: **Procedure for replacing paths in a portfolio** - For a given portfolio P, path $\underline{p}$ is the shortest path remaining in the portfolio after the optimal attack. The procedure replaces all paths in P that are longer than $\underline{p}$ with a copy of $\underline{p}$. For convenience, we use strong inequalities in the figure, which are not essential for the proof.

Lemma 1. *Given an operational portfolio P, the portfolio $P^R = \textsc{Replace}(P)$ is also operational, has the same optimal attack, and results in the same response path from a defender.*

Proof. Let E^* be the optimal attack for portfolio P. Let $\underline{p}$ be the shortest non-interdicted path in P under the optimal attack. Any other attack plan, $E \neq E^*$ leaves at least one non-interdicted path with shorter or equal length to that of $\underline{p}$. The Replace procedure only replaces paths in P that are longer than $\underline{p}$. Therefore, for any attack plan E, the shortest non-interdicted path is the same in P and P^R. Specifically, E^* is the optimal attack plan against portfolio P^R, and the defender's response remains unchanged. Since the optimal attack plan leaves at least one non-interdicted path, portfolio P^R is operational. □

For a portfolio P that contains multiple copies of a path, e.g., P' in Figure 2.1, one can find another portfolio that contains fewer paths, but performs as well as P. That is because copies of a path are unnecessary in the portfolio.

Lemma 2. *Let P^{MIP} be the portfolio that is the optimal solution for model (2.3). Then, the length of the shortest non-interdicted path in P^{MIP} under an optimal attack equals the length of the longest path in P^{MIP}.*

Proof. We prove the result by contradiction. Assume that the shortest non-interdicted path in P^{MIP} under an optimal attack is shorter than the longest path in P^{MIP}. Consider the portfolio $P^R = \text{Replace}(P^{MIP})$. The portfolio P^R is feasible in model (2.3), by Lemma 1, and has a smaller objective function value than P^{MIP} by the definition of the Replace procedure. In particular, the procedure replaces the longest path with a shorter path. Thus, P^{MIP} cannot be optimal for model (2.3), resulting in a contradiction. □

Theorem 1. *An optimal solution for model (2.3) is also an optimal solution for the equivalent PPS-m-n problem.*

Proof. Assume the optimal solution for the PPS-m-n problem is P^*. We show that model (2.3) returns a solution as least as good as P^*, measured by the length of the shortest non-interdicted path. We use the notation $\overline{z}(P)$ to denote the length of the longest path in P and $\underline{z}(P)$ to denote the length of the shortest non-interdicted path in P under an optimal attack. Let $P^R = \text{Replace}(P^*)$, we then have:

$$\underline{z}(P^*) \overset{(1)}{=} \underline{z}(P^R) \overset{(2)}{=} \overline{z}(P^R) \overset{(3)}{\geq} \overline{z}(P^{MIP}) \overset{(4)}{=} \underline{z}(P^{MIP})$$

Equalities (1) and (2) hold by Lemma 1. Inequality (3) holds because P^R is a feasible portfolio in model (2.3), and thus produces an objective function value larger than or equal to the optimal. Equality (4) holds because of Lemma 2, which completes the proof. $\square$

2.3.2 Row-and-column Generation Approach to Solve PPS-*m*-*n*

The number of attack plans considered in (2.3) grows as $\binom{|A|}{n}$. Each attack plan induces its own set of indicator variables ι_E^k, and generates its own set of constraints (2.3d)–(2.3f). Model (2.3) becomes intractable for larger networks because it is unreasonable to write out all the constraints. In this section, we show how a series of smaller optimization problems can be used to generate a set of attack plans that is significantly smaller than $\mathcal{E}$, but sufficiently large to find the optimal solution for PPS-m-n.

We call a model that incorporates all possible attack plans a *complete* model. Consider a version of model (2.3) that contains only an arbitrary subset of attack plans. We refer to such a model as *incomplete*. The benefit of the incomplete model is that it can have significantly fewer constraints. The drawback is that its solution is not necessarily feasible for the complete model. To find out if a portfolio is feasible, one needs to solve the attacker's optimization problem (2.4). If there exists an attack plan that interdicts all paths in the portfolio simultaneously, the optimal objective function value for (2.4) is greater than the value M. We can make two important statements in that case: (1) the portfolio is non-operational and infeasible for the complete model, and (2) the associated optimal attack plan is missing in the subset of attack plans we considered in the incomplete model.

With a row-and-column generation approach we sequentially solve incomplete models and attack the resulting portfolios. The outcome of the attack is used to augment the subset of attack plans, generating additional rows and columns in the next model to solve. If the optimal solution for an incomplete model is operational, there is no attack plan that interdicts all path in the portfolio, meaning that the portfolio is feasible and optimal for the complete model. It is optimal because the incomplete model is a relaxation of the complete model, which is a minimization problem. The idea of the algorithm is depicted in Figure 2.2.

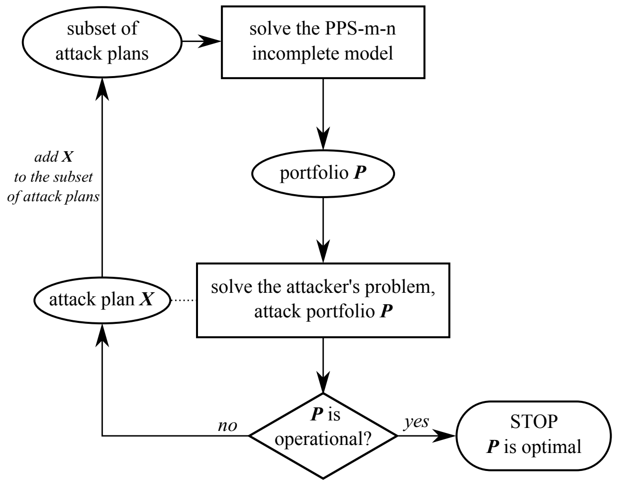

subset of attack plans
solve the PPS-m-n incomplete model
portfolio P
add X to the subset of attack plans
solve the attacker's problem, attack portfolio P
attack plan X
no
P is operational?
yes
STOP P is optimal

Figure 2.2: **Row-and-column generation procedure for PPS-m-n** - We start with a subset of attack plans and find the optimal solution P for the resulting incomplete model. The attack plan that renders P non-operational is added to the subset of attack plans. As soon as the incomplete model returns a P that is operational, we stop the procedure.

Solving the MIP for an incomplete program can be time consuming. Therefore, we are interested in keeping the number of iterations of the row-and-column generation procedure in Figure 2.2 low. We can improve the procedure in two places. First, rather than arbitrarily selecting a subset of attack plans, one can start with a reasonable subset of attack plans. We later discuss that the solution to SPNI is a lower bound for PPS-m-n. Solving the SPNI problem following Benders' decomposition approach provides an attack plan in every iteration, giving a reasonable subset of attack plans to start the row-and-column generation procedure. One can add more than one attack plan in every iteration. There are often many attack plans that render a portfolio non-operational. We could add all of the attack plans that render a candidate portfolio non-operational to the incomplete program. However, if a portfolio has many attack plans that render it non-operational, it may be computationally intractable to find and add all of them to the next incomplete model. We introduce *goodness* of a portfolio as a quality attribute that kind of measures the independence of the portfolio's paths. A portfolio is better if there exists fewer attack plans that render the portfolio non-operational. Generating good portfolios maintains tractability when adding attack plans to the incomplete program. To find good portfolios, one can add supervalid inequalities to the incomplete program, as described next.

2.3.3 Supervalid Inequalities and Model Enhancements

This section gives some additional constraints, supervalid inequalities, one can add to model (2.3). The overall purpose of these model extensions is to reduce the computational time required to find a solution within a desired optimality gap. The constraints cut suboptimal candidate solutions from the solution space, remove symmetry from the problem, and provide lower bounds. Another example that uses supervalid inequalities can be found in [7].

For an incomplete program, some of the extensions have an additional positive effect. The optimal solution for an incomplete model can be non-operational, providing attack plans that need to be added to the subset of attack plans for the next incomplete model to solve. Supervalid inequalities can help to produce better non-operational portfolios, where better means there are fewer attack plans that render the portfolio non-operational.

Lower bound for the PPS-m-n problem

MIPs are typically solved using branch-and-bound (B&B) techniques [26]. B&B enumerates candidate solutions, where parts of the candidates are removed from consideration after evaluating upper and lower bounds. In the worst-case scenario, B&B has to explore and evaluate each candidate solution in the branching tree in order to find the optimal solution or prove optimality for a feasible solution in hand. A tight lower bound can prevent the B&B algorithm from further exploring the B&B tree. A feasible solution with objective function value equal to a provided lower bound cannot be improved. Once the solver has found such a solution it can stop enumerating other solutions. Model (2.3) is a MIP, and in this section we provide a lower bound for PPS-m-n, which can be found quickly.

The solution to the Shortest-Path Network-Interdiction (SPNI) problem with n attacks is a lower bound for the PPS-m-n problem. The SPNI problem is a bi-level attacker-defender problem. The attacker has the ability to interdict n arcs in the network, such that the length of the shortest path in the interdicted network is maximized. The defender selects that shortest path. The primal formulation of the SPNI problem is shown in (2.5).

Primal formulation for SPNI

$$\max_{\boldsymbol{x}} \; \min_{\boldsymbol{y}} \sum_{(i,j)\in A} (c_{ij} + Mx_{ij}) y_{ij} \tag{2.5a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij} - \sum_{i:(j,i)\in A} y_{ji} = \begin{cases} -1 & \text{if } j = s & : [\pi_s] \\ 0 & \text{if } j \in N\backslash\{s,t\} & : [\pi_j] \\ 1 & \text{if } j = t & : [\pi_t] \end{cases} \tag{2.5b}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{2.5c}$$

$$\sum_{(i,j)\in A} x_{ij} \leq n \tag{2.5d}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.5e}$$

Standard solvers typically cannot handle this type of problem, but there are ways to reformulate it. One can yield a single-level maximization problem by taking the dual of the inner problem, here in terms of the dual variables π. A second way is to decompose the problem into master and sub-problem using Benders' decomposition. The full master, as shown in (2.6) is in terms of paths, rather than arcs as in the primal formulation. We highlight the similarity to the attacker's problem (2.4).

Full master for SPNI with Benders' decomposition

$$\max_{\boldsymbol{x},\, z_A} \; z_A \tag{2.6a}$$

$$\text{s.t. } z_A \leq \sum_{(i,j)\in\overline{\boldsymbol{y}}} (c_{ij} + Mx_{ij}) \qquad \forall \overline{\boldsymbol{y}} \in \overline{Y} \tag{2.6b}$$

$$\text{(2.5d), (2.5e)}$$

Lemma 3. *The solution for the SPNI problem is a lower bound for the PPS-m-n problem. Furthermore, the gap between the lower bound and PPS-m-n is greater than or equal to the gap between the lower bound and PPS-$(m+1)$-n. Finally, there is a value of m large enough so that the lower bound is equal to the optimal value of PPS-m-n.*

Proof. Assume a portfolio of large size, large enough to hold all simple paths from s to t in the network. The formulation for attacking the portfolio (2.4) is equal to the SPNI master problem (2.5), and hence, the optimal objective function values are the same. This gives an m large enough, so that the optimal value of PPS-m-n is equal to the lower bound. For smaller values of m, the PPS-m-n problem restricts the defender to a subset of all the available paths between s and t. This restriction cannot improve the objective function value, which proves that the solution for the SPNI problem gives a lower bound. Finally, $m+1$ paths in the portfolio can only perform better or equal to m paths, completing the proof of the lemma. $\square$

Let LB be the value of the lower bound, which can be computed quickly even for large networks. One supervalid inequality we can add to model (2.3) is

$$\overline{z} \geq LB. \tag{2.7}$$

The advantage of this supervalid inequality is that if m is large enough to match the lower bound, and we start with the attack plans from solving SPNI using Benders' decomposition, we can achieve an optimal solution for model (2.3) in very few iterations of the row-and-column generation procedure, as depicted in Figure 2.2. The impact of the portfolio size to the optimal solution is shown as an example in Section 2.5.1.

Restriction on arcs appearing too often

To enforce a certain level of path independence in the portfolio, we add constraints that prohibit any arc in the network to be part of more that $m-n$ paths in the portfolio. If an arc is part of more than $m-n$ paths, then the attacker can interdict that arc, and use the remaining interdiction budget of $n-1$ to interdict each of the remaining paths individually. This produces an attack plan that interdicts all the paths in the portfolio, meaning that the portfolio is non-operational and infeasible in the complete model. A supervalid constraint of type (2.8) prevents arcs from appearing in too many paths in the portfolio. Adding such a supervalid constraint to an incomplete model generates better portfolios as the row-and-column generation procedure in Figure 2.2 executes, reducing the number of required iterations and the number of constraints added in each iteration.

$$\sum_{k \in K} y_{ij}^{k} \leq m-n \qquad \forall (i,j) \in A \tag{2.8}$$

Minimal cutset for the portfolio

A portfolio can be seen as a subgraph of G. Each arc that is part of at least one path in the portfolio is also part of the subgraph. The subgraph has a minimum cardinality s-t cut. If the cardinality of this cutset is smaller than n, the attacker's budget suffices to disconnect s from t. That means the attacker can interdict each path in the portfolio that formed the subgraph using no more than n attacks. To exclude such portfolios from consideration one can add the constraints shown in (2.9). Such supervalid constraints eliminate many non-operational portfolios from consideration, and can significantly improve the solutions seen from incomplete models.

Additional notation

f_{ij}	flow on arc (i, j) in the subgraph formed by the portfolio P
v	maximum flow that can be pushed from s to t in the subgraph, assuming that each arc can carry at most one unit of flow

Formulation for cutset constraints

$$\sum_{i:(i,j)\in A} f_{ij} - \sum_{i:(j,i)\in A} f_{ji} = \begin{cases} -v & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ v & \text{if } j = t \end{cases} \tag{2.9a}$$

$$v \geq n + 1 \tag{2.9b}$$

$$f_{ij} \leq \sum_{k\in K} y_{ij}^k \qquad \forall (i,j) \in A \tag{2.9c}$$

$$f_{ij} \leq 1 \qquad \forall (i,j) \in A \tag{2.9d}$$

The well-known duality between minimum cut and maximum flow problem enables us to formulate constraints (2.9) like a maximum flow problem. The classic maximum flow problem contains flow conservation constraints (2.9a), with v as objective function one wish to maximize. Constraint (2.9b) requires v, the maximum flow in the subgraph, to be greater than the attacker's budget. Each arc in the subgraph can carry at most one unit of flow by (2.9d). The link to the subgraph is built by (2.9c). Only arcs that are part of at least one path in the portfolio can carry flow. The flow variables f_{ij} are forced to be zero if the arc (i, j) is not present in the portfolio.

Path ordering according to path length

PPS-m-n is highly symmetric. As mentioned, (2.3) is a multicommodity flow problem, with one commodity for each path. The assignment of commodity number k to a path in the portfolio is unimportant. Given a feasible portfolio with m different paths, one can find $m!-1$ additional equally good feasible solutions. This symmetry can significantly increase computational time when searching for an optimal solution. Ordering the paths in the portfolio according to path length removes some of this symmetry. The constraint to order the paths is

$$\sum_{(i,j)\in A} c_{ij} y_{ij}^{k} \leq \sum_{(i,j)\in A} c_{ij} y_{ij}^{k+1} \qquad \forall k = 1, \ldots, m-1. \tag{2.10}$$

Subtour elimination

Technically speaking, the solution for model (2.3) is a portfolio of walks. A walk is a sequence of arcs that connects s and t, and could contain a cycle, often called a subtour. We adopt the terminology "subtour" and "subtour elimination" from the traveling salesman problem (TSP). Literature calls a walk that does not contain a cycle a path [19]. The objective function in (2.3) guarantees that the longest path in the optimal solution is actually a path. All the other entries in the portfolio could be walks, because they can contain subtours in any feasible solution. In PPS-m-n we are concerned about paths, and one can easily remove subtours from the optimal solution without affecting feasibility or optimality of the portfolio. However, including subtour elimination constraints can help exclude candidate solutions from consideration and therefore decrease runtime. Subtour elimination constraints have been developed for TSP [27], for which a walk is not a solution. PPS-m-n does not need subtour elimination constraints to generate an optimal portfolio.

We distinguish two different kinds of subtours. The first type we call *loops*, which is a subtour that involves nodes which are also on the path from s to t. The second type we call *circles*, which do not involve nodes that are on the path from s to t. One way to formulate subtour elimination constraints can be found in [27]. Adding such constraints as shown in (2.11) will eliminate both kinds of subtours.

Additional notation

u_i^k	a label for node i associated with path $k \in K$

Formulation for subtour elimination constraints

$$u_s^k = 1 \qquad \forall k \in K \tag{2.11a}$$

$$u_i^k - u_j^k + 1 \leq (|N| - 1)(1 - y_{ij}^k) \qquad \forall k \in K,\ (i,j) \in A \tag{2.11b}$$

$$1 \leq u_i^k \leq |N| \qquad \forall k \in K,\ i \in N \tag{2.11c}$$

The idea behind the formulation is to label the nodes for each path according to the following labeling rules: (1) the source is labeled with one for each path in the portfolio (2.11a), (2) following each path from source to sink, the nodes are labeled in ascending order and adjacent nodes in a path need to have consecutive labels (2.11b). If a path contains a subtour one cannot complete such a labeling. Vice versa, a path satisfying the constraints (2.11) cannot contain a subtour.

The main reason for eliminating subtours is to cut runtime. Adding constraints and variables as (2.11) to the model excludes subtours, but also increases model size. Simpler subtour elimination constraints are shown in (2.12). The constraints do not rely on new variables, decreasing model size. If each node in the path has at most one outgoing arc, the path from s to t cannot contain a loop. The drawback is that the solution can very well contain circles. However, we can detect an remove circles from the optimal solution, after the solution process.

$$\sum_{j:(i,j)\in A} y_{ij}^k \leq 1 \qquad \forall k \in K,\ i \in N \tag{2.12}$$

2.3.4 Special Cases of PPS-m-n

The PPS-m-1 problem

The attacker in PPS-m-1 is restricted to attack a single arc in the network. This special instance of PPS-m-n can be formulated as a MIP with a reduced set of constraints. The model for PPS-m-1 can be solved directly, meaning that it is unnecessary to apply the row-and-column generation approach described in Section 2.3.2.

We restate the set of constraints containing the variables ι_E^k from model (2.3). In this special case the variables ι_E^k are not required at all, and their constraints can be substituted by a single constraint. Specifically, constraints

$$\begin{aligned} \iota_E^k &\geq y_{ij}^k && \forall k \in K,\ (i,j) \in E,\ E \in \mathcal{E} && \text{(2.3d)} \\ \iota_E^k &\leq \sum_{(i,j)\in E} y_{ij}^k && \forall k \in K,\ E \in \mathcal{E} && \text{(2.3e)} \\ \iota_E^k &\leq 1 && \forall k \in K,\ E \in \mathcal{E} && \text{(2.3f)} \\ \sum_{k\in K} \iota_E^k &\leq m-1 && \forall E \in \mathcal{E} && \text{(2.3g)} \end{aligned}$$

can be substituted with

$$\sum_{k\in K} y_{ij}^k \leq m-1 \qquad \forall (i,j) \in A. \tag{2.13}$$

An attack plan consists of a single arc, $|E| = 1$ and $\iota_E^k \triangleq \iota_{ij}^k$. Constraints (2.3d)–(2.3e) set $\iota_{ij}^k = y_{ij}^k$, just another notation for the same variable. We substitute ι_E^k with y_{ij}^k in the formulation, which makes all but constraints (2.3g) redundant. Constraint (2.13) is simply a re-statement of constraint (2.3g) for the special case.

The PPS-m-$(m-1)$ problem

The PPS-m-n model does not require a portfolio of independent paths as a solution, but as the number of attacks approaches the number of paths, the resulting optimal portfolios eventually become more independent. The solution to PPS-m-$(m-1)$ is a set of m edge-disjoint paths; otherwise the attack budget is sufficent to render the portfolio non-operational. The disjointness can be enforced by using constraint (2.8), because m–n=m–(m–1) = 1. Any portfolio of disjoint paths is operational, rendering constraints (2.3d)–(2.3g) unnecessary.

2.3.5 Problem Bounding and Network Manipulation

Model (2.3) grows quickly in network size. Even an incomplete model, as used for the row-and-column generation approach of Section 2.3.2, can become intractable. In this section, we discuss how to shrink a large network, without affecting the optimal solution. In other words, we remove parts from the network that are unnecessary to obtain the optimal solution. Shrinking the network leads to models with fewer variables and constraints, and reduced solution time.

Formulation (2.3) is a minimization problem, and any feasible solution provides an upper bound for the objective function value. The objective function is linked to the longest path in the portfolio. Hence, the longest path in any feasible portfolio is an upper bound for the longest path in the optimal portfolio. We can make use of this fact and exclude all paths from consideration that are longer than an upper bound in hand. None of those paths can be part of the optimal portfolio. Formulation (2.3) is in terms of nodes and arcs and not in terms of paths. Therefore, we use the upper bound to remove unnecessary nodes and arcs from the network: nodes and arcs that cannot be in any of the optimal portfolio's paths.

Let $\text{spl}(i, j)$ be the shortest path length from nodes i to node j in the network. We can find $\text{spl}(s, i)$ and $\text{spl}(i, t)$ for all nodes i in the network efficiently, i.e., applying Dijkstra's algorithm on the original and reversed network, respectively. For each node i, we also calculate the length of the shortest path from s to t that includes node i, $\text{spl}(s, i)+\text{spl}(i, t)$. Assume a feasible portfolio with a longest path length, $\overline{z}$. If $\text{spl}(s, i) + \text{spl}(i, t) > \overline{z}$, than a portfolio containing node i in any path cannot be optimal. We can remove node i and all incident arcs from the network. The same argument holds considering arcs. If $\text{spl}(s, i) + c_{ij} + \text{spl}(j, t) > \overline{z}$, arc (i, j) can be removed from the network.

Removing nodes and arcs as described can be done efficiently, if we have a feasible portfolio that provides a good upper bound. The tighter the upper bound, the larger the reduction in network size. We introduce three ways of computing upper bounds. Each increases the computational complexity of the previous; however, each also yields a tighter upper bound than the previous. In practice, we shrink the network using the loosest, computationally cheapest upper bound first. Then, on the decreased network, we compute a more expensive but tighter upper bound to decrease network size further, and so on.

A portfolio of n+1 disjoint paths

A portfolio of n+1 disjoint paths is clearly operational, since an attack budget of n cannot destroy all paths in P. A feasible portfolio of m paths for model (2.3) can be constructed by duplicating some of the disjoint paths, providing an upper bound. We find two different sets of n+1 disjoint paths.

First, one can find the set of disjoint paths sequentially. We repeatedly identify and remove the shortest path in the network. The length of the shortest path in the n+1 iteration is the desired upper bound. This bound is unlikely to be tight but typically enables some network reduction. The network reductions can significantly reduce the runtime to find a optimized set of n+1 disjoint paths, which provides the next better bound.

The linear program shown in (2.14) is similar to PPS-m-(m−1), and returns a set of k disjoint paths, such that the longest path has minimun length. No matter the portfolio size m in the original PPS-m-n problem, we seek to bound, here we set $K = \{1, \ldots, n+1\}$.

$$\min_{\boldsymbol{y}, \overline{z}} \quad \overline{z} \tag{2.14a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij}^k - \sum_{i:(j,i)\in A} y_{ji}^k = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N \backslash \{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall k \in K \tag{2.14b}$$

$$\overline{z} \geq \sum_{(i,j)\in A} c_{ij} y_{ij}^k \qquad \forall k \in K \tag{2.14c}$$

$$\sum_{k\in K} y_{ij}^k \leq 1 \qquad \forall (i,j) \in A \tag{2.14d}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ (i,j) \in A \tag{2.14e}$$

Solving (2.14) is harder than sequentially finding shortest paths in a network, although it yields a potentially tighter bound.

A portfolio of m paths with arc use constraints

For this bound, we produce a portfolio of m paths. The attacker can interdict up to m−1 paths in the portfolio for the portfolio to be operational. If no arc appears in more than $\lfloor \frac{m-1}{n} \rfloor$ paths, then the portfolio is operational. This is because with each attack, the attacker can interdict at most $\lfloor \frac{m-1}{n} \rfloor$ paths, for a total of $\lfloor \frac{m-1}{n} \rfloor \cdot n \leq m-1$ paths. Solving model (2.14) for m paths with constraint (2.14d) replaced by

$$\sum_{k \in K} y_{ij}^k \leq \left\lfloor \frac{m-1}{n} \right\rfloor \qquad \forall (i,j) \in A \tag{2.15}$$

yields such a feasible portfolio.

The two portfolios consisting of disjoint paths, as well as the solutions that limit the use of any arc in the portfolio, are specific feasible solutions for PPS-m-n. The associated models are simple, quick to solve, and provide good bounds. However, any feasible solution provides an upper bound on the length of the longest path in P and can be used to shrink the network. So, potentially, one could implement a custom B&B that continuously cuts the network as the solution tree is explored.

Further network simplifications

After bounding the problem and reducing the network, one should consider further network simplifications before solving (2.3). There are potentially more nodes and arcs in the network that can be removed, saving decision variables in the model. Nodes that appear in any path from source to sink need to have at least an incoming arc and outgoing arc. For that reason, nodes as u and w in Figure 2.3 cannot appear in the portfolio. The only successor of node v in Figure 2.3 is also the only predecessor of v. Node v can be removed from the network because any path containing v also contains a loop. Finally, consider a node j with only one incoming arc (i,j) and several outgoing arcs. Arc (j,i) can be removed from the network because this is an arc that, if included in a path, creates a loop or a cycle. Arc e in Figure 2.3 is such an arc. The same argument holds for a node with only one outgoing arc (i,j) and several incoming arcs. Arc (j,i) can be removed if it exists.

Furthermore, for arcs (u,v) and (v,w) in Figure 2.4 either both appear or neither appears in a path. One can replace the two arcs (u,v) and (v,w) with the new arc (u,w), if such an arc does not exist already, and save a decision variable in the model. Arc (u,w) has to be re-substituted if it appears in the solution because it is not present in the original network.

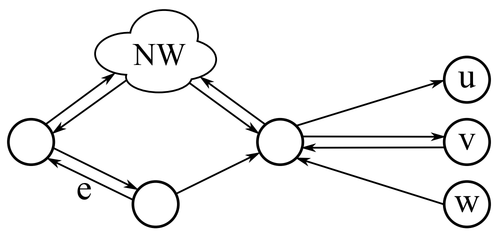


Figure 2.3: **Removing unnecessary nodes and arcs** - All nodes but s and t with only one incident arc can be removed from the network because they can not appear in a complete s-t path, e.g., nodes u and w. The appearance of node v or arc e in a path creates a loop or cycle. Loops and cycles can be removed, and therefore, v and e can be removed from the network.

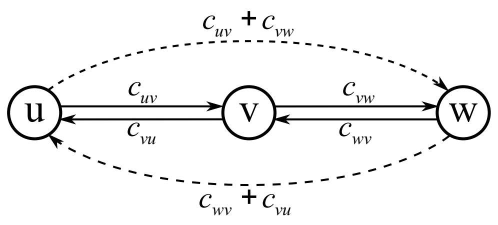


Figure 2.4: **Additional network simplifications** - In this example, if the network consists solely of the solid edges, the only way of reaching node w from node u is by visiting node v. If arc (u, v) is in one of the portfolio paths, arc (v, w) must be in the path as well. One can add the new arcs (u, w) and (w, u) with their associated lengths and remove node v from the network. Solutions from the simplified network can be translated to the original network directly.

Section 2.5.2 contains two examples with large networks. We use the bounding and network shrinking approach as described and solve PPS-5-2 for each network. For a road network of Germany, we plot the intermediate solutions for the bounding optimization models and the network after the shrinking process.

2.3.6 Pr-PPS-m-n with Attack Probabilities

We can alter PPS-m-n slightly by incorporating a probability that an attack will happen, resulting in the *Probabilistic-Path-Portfolio-Selection-m-n* problem, Pr-PPS-m-n. Similar attack probabilities have been added to standard network-interdiction models [28, 29, 30, 31]. For the special cases from Section 2.3.4, we can find optimal portfolios that consider the likelihood of an attack. For the general case, we present a model that is correct but is intractable for larger networks. However, we also present a method that finds good, but not optimal, solutions quickly.

To select a portfolio that accounts for the probability of an attack, one has to change the objective function of model (2.3). The length of the longest path should be replaced by the expected length of the defender's response, where the expectation is taken over the attack probabilities. This requires knowing the length of the response path after ℓ attacks for $\ell = 0, \ldots, n$. The proofs in Section 2.3.1 show that the longest path is the defender's response when ℓ equals n. However, for other values of ℓ, it is less clear which path is selected by the defender. We add constraints to the model that calculate the optimal response for the defender after ℓ attacks.

Additional notation

p_ℓ	probability that the attacker interdicts ℓ arcs in the network, with $0 \le \ell \le n$, and $\sum\limits_{\ell=0}^{n} p_\ell = 1$
$\underline{z}_\ell$	length of the shortest path in the portfolio after an attack on ℓ arcs, $0 \le \ell \le n$
$E_\ell \in \mathcal{E}_n$	$\mathcal{E}_n$ is the set of all cardinality ℓ subsets of A, with $0 \le \ell \le n$, representing all possible attack plans; the attacker interdicts all arcs in E_ℓ for a specific attack plan $E_\ell \in \mathcal{E}_n$
$\iota^k_{E_\ell}$	$\iota^k_{E_\ell} = 1$ if attack plan E_ℓ interdicts path k, and $\iota^k_{E_\ell} = 0$ otherwise; in other words, $\iota^k_{E_\ell} = 1$ if at least one $(i,j) \in E_\ell$ is in path k

Pr-PPS-m-n model

$$\min_{\boldsymbol{y},\,\boldsymbol{\iota},\,\underline{\boldsymbol{z}}} \quad \sum_{\ell=0}^{n} p_\ell \cdot \underline{z}_\ell \tag{2.16a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij}^k - \sum_{i:(j,i)\in A} y_{ji}^k = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N \backslash \{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall k \in K \tag{2.16b}$$

$$\sum_{\forall (i,j)\in A} c_{ij} y_{ij}^k \le \sum_{\forall (i,j)\in A} c_{ij} y_{ij}^{k+1} \qquad \forall k = 1, \ldots, m-1 \tag{2.16c}$$

$$\iota_{E_\ell}^k \ge y_{ij}^k \qquad \forall k \in K,\ (i,j) \in E_\ell,\ E_\ell \in \mathcal{E}_n \tag{2.16d}$$

$$\iota_{E_\ell}^k \le \sum_{(i,j)\in E_\ell} y_{ij}^k \qquad \forall k \in K,\ E_\ell \in \mathcal{E}_n \tag{2.16e}$$

$$\iota_{E_\ell}^k \le 1 \qquad \forall k \in K,\ E_\ell \in \mathcal{E}_n \tag{2.16f}$$

$$\sum_{k\in K} \iota_{E_n}^k \le m-1 \qquad \forall E_n \in \mathcal{E}_n \tag{2.16g}$$

$$\underline{z}_\ell \ge \sum_{(i,j)\in A} c_{ij} y_{ij}^k - M \iota_{E_\ell}^k - M \sum_{k'<k} (1 - \iota_{E_\ell}^{k'})$$
$$\forall \ell = 0, \ldots, n,\ k \in K,\ E_\ell \in \mathcal{E}_n \tag{2.16h}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ (i,j) \in A \tag{2.16i}$$

Constraints (2.16b) and (2.16c) enforce a portfolio of m paths which is ordered according to path length. Similar to model (2.3), constraints (2.16d)–(2.16f) flag a path if interdicted by an specific attack plan. Here, we consider attack plans with different numbers of attacked arcs. The indicator variable $\iota_{E_\ell}^k$ is forced to one if attack plan E_ℓ interdicts path k in the portfolio by (2.16d) and (2.16f). The indicator takes on the value zero if path k and attack plan E_ℓ do not have an arc in common by (2.16e). Constraint (2.16g) ensures an operational portfolio, because no attack plan of cardinality n attacks all paths in the portfolio simultaneously. The defender's response is the shortest non-interdicted path. Constraint (2.16h) selects the shortest path length in the interdicted portfolio, which is used to calculate the expected response length (2.16a). Constraint (2.16h) is vacuous for any interdicted path, or any path that has a shorter non-interdicted path.

Model (2.16) contains a huge number of constraints, which makes it intractable. An incomplete model that considers only a subset of attack plans can be significantly smaller. Solving an incomplete model gives a lower bound to the complete model because we solve a relaxation of the complete model. In contrast to PPS-m-n, the first operational portfolio found in the row-and-column generation procedure is not necessarily optimal, but it allows us to compute an upper bound on the optimal solution value. The defender's response to an attack that interdicts ℓ arcs can be found by solving the attacker's optimization problem (2.4) with an attack budget of ℓ. Given an operational portfolio, solving the attacker's optimization problem for all ℓ and computing the expected length of the defender's response gives an upper bound to the complete model. When the upper and lower bounds for the complete model are equal, we know the portfolio that yielded the upper bound is optimal. On the other hand, when the bounds are not equal, for some value of ℓ, the incomplete program is missing the optimal attack plan against the portfolio. We add the optimal attack plans for all ℓ that are not already included in the incomplete model, which generates new constraints in the next incomplete model to solve. The row-and-column generation procedure is depicted in Figure 2.5. An example comparing the solutions for PPS-m-n and Pr-PPS-m-n on a small network is shown in Section 2.5.3.

The Pr-PPS-m-1 problem

For this special case, the attacker interdicts either one or zero arcs. The defender in the Pr-PPS-m-1 model responds to the optimal attack with the longest path in P. If there is no attack, the defender responds with the shortest path in P. If there is uncertainty about the attack the defender seeks a portfolio that has minimum length for both, the shortest and longest path in the portfolio, weighted by the likelihood of attack. We can formulate the problem as follows:

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & p_0 \cdot \sum_{(i,j)\in A} c_{ij} y_{ij}^1 + p_1 \cdot \sum_{(i,j)\in A} c_{ij} y_{ij}^m \\ \text{s.t.} \quad & (2.3\text{b}), (2.3\text{h}), (2.10), (2.13). \end{aligned} \tag{2.17}$$

The sum in the objective function (2.17) is the expected length of the shortest path in the portfolio. With probability p_0 there is no attack and the shortest path in the

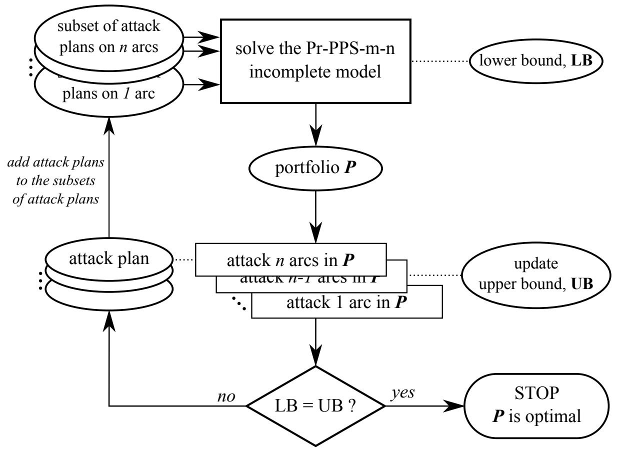


Figure 2.5: **Row-and-column generation procedure for Pr-PPS-m-n** - We start with a subset of attack plans and find the optimal portfolio P for the resulting incomplete model. That solution is a lower bound. We directly attack P with an attack budget ℓ, for $\ell = 1, \ldots, n$, and compute the expected response length. This value is used to update the upper bound. From all the portfolios we encountered so far, the defender will choose the one with shortest expected response length. Once upper and lower bound are equal, the portfolio that gave the upper bound is optimal. If P is suboptimal, we add all optimal attack plans E_ℓ to the subset of attack plans. Each E_ℓ is found by solving the attacker's optimization model (2.4) with different attack budget.

portfolio is used, and with probability p_1 there is one attack, and the longest path in the portfolio is used. To complete the model we add the flow conservation constraints for each path in the portfolio (2.3b), the binary property of the decision variables (2.3h) and the constraint (2.13) that limit the appearance of each arc in the portfolio to $m-1$. The constraint (2.10) that orders the path in the portfolio according to length is essential for the model. It forces the shortest path to appear at position $k = 1$ and the longest path at position $k = m$ in the portfolio.

The Pr-PPS-m-$(m-1)$ problem

The solution for PPS-m-$(m-1)$ is a portfolio of m disjoint paths. Each of the $m-1$ attacks can interdict at most one path in the portfolio. The smart attacker interdicts the paths in order of path length, starting with the shortest, until the attack budget is exhausted. The shortest non-interdicted path is the defender's response to the attack. Ordering the paths according to length allows us to formulate the problem of selecting an optimal portfolio while at the same time incorporating attack probabilities as:

$$\min_{\boldsymbol{y}} \quad \sum_{k \in K} \left(p_{(k-1)} \sum_{(i,j) \in A} c_{ij} y_{ij}^{k} \right) \tag{2.18}$$
$$\text{s.t.} \quad (2.3\text{b}),\ (2.3\text{h}),\ (2.8), (2.10)$$

The objective function (2.18) is the expected length of the remaining shortest path after an attack. With probability $p_{(k-1)}$, for $k = 1 \ldots m-1$, exactly $k-1$ attacks occur leading the defender to choose the k^{th} shortest path in the portfolio. The constraints completing the model are the flow conservation constraints for each path (2.3b) and the binary property of the decision variables (2.3h). The paths are disjoint by constraint (2.8), and ordered according to path length through constraint (2.10).

An upper bound for Pr-PPS-*m*-*n*

Solving for the optimal portfolio with attack probabilities in the general Pr-PPS-m-n problem is computationally difficult. The difficulty comes from determining the defender's response to any attack on ℓ arcs with $0 < \ell < n$ in model (2.16). The easier model (2.3) is unable to determine the length of the defender's response to the attack. However, we can find an upper bound on this length in model (2.3).

Lemma 4. *Let P be the optimal solution for PPS-m-n, and let z_k be the length of the k^{th} shortest path in P, and $\underline{z}_\ell$ the length of the defender's response path to an optimal attack on ℓ arcs, then $\underline{z}_\ell \leq z_{m-n+\ell}$ for $0 \leq \ell \leq n$.*

Proof. The statement is true for ℓ equals n by Theorem 1. Assume there exists an $\ell < n$ such that $\underline{z}_\ell > z_{m-n+\ell}$. That means, the optimal ℓ attacks destroy all the $m-n+\ell$ shortest paths in the portfolio. We can augment those ℓ attacks with an additional attack targeting each of the remaining non-interdicted paths, to create a plan with n attacks that destroys the entire portfolio. Thus, if such an ℓ exists, the portfolio must be non-operational, contradicting the assumption that it is the optimal solution to PPS-m-n. $\square$

We can bound the expected length of the defender's response using three facts: (1) if there is no attack, the defender responds with the shortest path in the portfolio, (2) the upper bounds from Lemma 4 for $0 < \ell < n$, and (3) if there are n attacks, the defender responds with the longest path in the portfolio by Theorem 1. This leads to the following model, which minimizes the bound on the expected response:

$$\min_{\boldsymbol{y}} \quad \sum_{\ell=1}^{n} \left(p_\ell \sum_{(i,j)\in A} c_{ij} y_{ij}^{m-n+\ell} \right) + p_0 \sum_{(i,j)\in A} c_{ij} y_{ij}^{1} \tag{2.19}$$
$$\text{s.t.} \quad (2.3b),\ (2.3d),\ (2.3e),\ (2.3f),\ (2.3g),\ (2.3h),\ (2.10),$$

which is similar to model (2.3). The optimal objective function value of model (2.19) gives an upper bound to the optimal value of model (2.16). However, a defender using the optimal portfolio resulting from model (2.19) can in fact perform better than the model's objective function suggests. How well the defender performs can be computed after attacking the portfolio directly to find the true responses for $0 < \ell < n$.

Bounding Pr-PPS-*m*-*n* and shrinking the network

Shrinking the network can significantly decrease the solution time for PPS-m-n, and we desire to use the shrinking method as described in Section 2.3.5 for Pr-PPS-m-n. An upper bound on the paths length is retrieved from any feasible solution. Solutions for the bounding models from Section 2.3.5 and the optimal solution for PPS-m-n are feasible for the Pr-PPS-m-n model. However, better bounds can also be achieved by substituting the objective function and path ordering constraints from model (2.19) into the bounding models for PPS-m-n. The actual bound on the response length is computed as follows.

Let z be an upper bound for the objective function value of Pr-PPS-m-n, let P^* be the optimal portfolio for Pr-PPS-m-n with expected response length z^*, and let *spl* be the length of the shortest path in the network. In addition, we denote by $l(k)$ the length of the k^{th} shortest path in a portfolio. We have

$$p_n \cdot l(m) + (1 - p_n) \cdot spl \ \le \ z^* \ \le \ z,$$

where the left hand side is an optimistic calculation of the expected response length. The defender's response to n attacks is $l(m)$, Lemma 2, and we assume an optimistic response length of *spl* for every other number of attacks. The second inequality holds because z is an upper bound on the optimal solution. Solving the above inequality for $l(m)$ gives $l(m) \le \frac{z-(1-p_n)spl}{p_n}$. A portfolio containing a path longer than $l(m)$ has an expected response length that is greater than z, and is therefore suboptimal. Hence $l(m)$ is an upper bound. An example for Pr-PPS-m-n using the bounding and shrinking procedure on a large network is shown in Section 2.5.3.

2.3.7 D-PPS-m-n with General Attacks

PPS-m-n assumes that an attacked arc is destroyed, and not usable by the defender. In some applications, attacking an arc may simply lengthen it as opposed to destroying it. In the introductory example, accepting a delay that is caused by clearing a roadblock may be shorter than bypassing the roadblock. In *Delayed-Path-Portfolio-Selection-m-n* problem, D-PPS-m-n, an attacked arc (i, j) is still usable for the defender, but lengthened by the fixed length d_{ij}. We introduce additional notation and formulate D-PPS-m-n as follows:

Additional notation

z_E^k	z_E^k is the length of path k in the portfolio after the attack E
$h_E^{k,k'}$	$h_E^{k,k'} = 1$, if path k is longer than path k' in the portfolio after attack E, and $h_E^{k,k'} = 0$, otherwise
d_{ij}	length added to the nominal length c_{ij} if arc (i, j) is attacked

D-PPS-m-n model

$$\min_{\boldsymbol{z},\boldsymbol{h},\boldsymbol{y},\underline{z}} \underline{z} \tag{2.20a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij}^k - \sum_{i:(j,i)\in A} y_{ji}^k = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall k \in K \tag{2.20b}$$

$$z_E^k = \sum_{(i,j)\in A} c_{ij} y_{ij}^k + \sum_{(i,j)\in E} d_{ij} y_{ij}^k \qquad \forall k \in K,\ E \in \mathcal{E} \tag{2.20c}$$

$$\underline{z} \geq z_E^k - M \sum_{\substack{k'\in K,\\ k'\neq k}} h_E^{k,k'} \qquad \forall k \in K,\ E \in \mathcal{E} \tag{2.20d}$$

$$M \cdot h_E^{k,k'} \geq z_E^k - z_E^{k'} \qquad \forall k, k' \in K,\ k \neq k',\ E \in \mathcal{E} \tag{2.20e}$$

$$h_E^{k,k'} + h_E^{k',k} = 1 \qquad \forall k, k' \in K,\ k \neq k',\ E \in \mathcal{E} \tag{2.20f}$$

$$h_E^{k,k'} + h_E^{k',k''} - 1 \geq h_E^{k,k''} \qquad \forall k, k', k'' \in K,\ k \neq k' \neq k'',\ E \in \mathcal{E} \tag{2.20g}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ (i,j) \in A \tag{2.20h}$$

$$h_E^{k,k'} \in \{0,1\} \qquad \forall k, k' \in K,\ E \in \mathcal{E} \tag{2.20i}$$

Constraint (2.20b) enforces selection of m paths from s to t. The path length under a given attack E is computed by (2.20c). For a specific attack plan, E, the right hand side of (2.20d) makes the constraint vacuous for any path that is not shortest in the attacked portfolio. There is only one non-vacuous constraint of type (2.20d) for each attack plan E, the one for the defender's best response. All of the non-vacuous constraints, together with the objective function, ensure that $\underline{z}$ equals the length of the defender's response to the worst-case attack. A value for big-M that is greater than the length of the longest interdicted path in the network is sufficiently large to render the appropriate constraints vacuous. If path k is longer than path k' under attack E, then $h_E^{k,k'}$ is forced to 1 by (2.20e), and $h_E^{k',k}$ is 0 by (2.20f). In case of a tie, one path is defined to be longer by (2.20f). If there are more than two paths of equal length, the assignment of which one is longer needs to satisfy transitivity. Such an assignment is enforced by (2.20g).

The set of constraints (2.20e)–(2.20g) contains redundancy that can be removed in favor of runtime. Because addition is commutative it is sufficient to build constraint (2.20f) for all $k < k'$ only. To ensure transitivity it is sufficient to include constraints (2.20g) for all k, k', k'' such that $k < k', k < k''$ and $k' \neq k''$.

Model (2.20) can be enhanced with attack probabilities, similar to model (2.16), and solved using the same row-and-column generation approach as described in Section 2.3.6. Any solution to an incomplete model provides a lower bound. The upper bound can be updated after attacking the solution to an incomplete model. The attacker's optimization problem is similar to (2.4), where the value for M is replaced by the associated d_{ij} of the attacked arc. An example for D-PPS-m-n is shown in Section 2.5.4.

2.3.8 C-PPS-m-n with Continuous Attack Variables

This section considers a problem where the attack budget is a total length the attacker can add to the network, spread continuously across all arcs. In particular, the attack budget can be spread continuously across the portfolio's paths, forcing the defender to respond with an interdicted path. When using an attacked arc, the defender accepts the additional length that is added to the arc. We introduce new variables, whose meanings are depicted in Figure 2.6, and formulate the *Continuous-Path-Portfolio-Selection-m-n* problem, C-PPS-m-n, as follows:

Additional notation

σ^k	flow that leaves the source on path k in the portfolio
τ^k	flow that arrives at the sink on path k in the portfolio
w^k_{ij}	flow on arc (i,j) on path k in the portfolio
x_{ij}	length added to the nominal length c_{ij} if arc (i,j) is attacked

C-PPS-*m*-*n* model, primal formulation

$$\min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \min_{\boldsymbol{w},\boldsymbol{\sigma},\boldsymbol{\tau}} \sum_{k\in K} \sum_{(i,j)\in A} w^k_{ij}(c_{ij}+x_{ij}) \tag{2.21a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y^k_{ij} - \sum_{i:(j,i)\in A} y^k_{ji} = \begin{cases} -1 & \text{if } j=s \\ 0 & \text{if } j\in N\backslash\{s,t\} \\ 1 & \text{if } j=t \end{cases} \qquad \forall k\in K \tag{2.21b}$$

$$\sum_{i:(i,j)\in A} w^k_{ij} - \sum_{i:(j,i)\in A} w^k_{ji} = \begin{cases} -\sigma^k & \text{if } j=s & :\left[\pi^k_s\right] \\ 0 & \text{if } j\in N\backslash\{s,t\} \quad \forall k\in K & :\left[\pi^k_j\right] \\ \tau^k & \text{if } j=t & :\left[\pi^k_t\right] \end{cases} \tag{2.21c}$$

$$\sum_{k\in K} \sigma^k = 1 \qquad :[\theta_s] \tag{2.21d}$$

$$\sum_{k\in K} \tau^k = 1 \qquad :[\theta_t] \tag{2.21e}$$

$$w^k_{ij} \le y^k_{ij} \qquad \forall k\in K,\ (i,j)\in A \qquad :\left[\rho^k_{ij}\right] \tag{2.21f}$$

$$w^k_{ij} \ge 0 \qquad \forall k\in K,\ (i,j)\in A \tag{2.21g}$$

$$\sigma^k, \tau^k \ge 0 \qquad \forall k\in K \tag{2.21h}$$

$$y^k_{ij} \in \{0,1\} \qquad \forall k\in K,\ (i,j)\in A \tag{2.21i}$$

$$\sum_{(i,j)\in A} x_{ij} \le n \tag{2.21j}$$

$$x_{ij} \ge 0 \qquad \forall (i,j)\in A \tag{2.21k}$$

Imagine a stacked network as shown in Figure 2.6. Constraint (2.21b) enforces the selection of a s-t-path in each of the m layers, representing the portfolio. Constraint (2.21d) creates one unit of flow to be sent out of the artificial node s'. Node t' demands one unit of flow by constraint 2.21e. A feasible solution satisfies the demand with the supply and appropriately routes the flow from s' to t', where only arcs in the portfolio's paths can carry flow by (2.21f). The flow conservation constraint from s' to t' is split into constraints (2.21c)–(2.21e). Pushing flow w_{ij}^k through arc (i, j) causes the associated cost c_{ij}, and in order to minimize the objective function value (2.21a), the flow is pushed across the shortest remaining path in the portfolio after the attack. The attacker's ability to enlarge arcs is limited by the budget constraint (2.21j). Only the y-variables are required to be binary by (2.21i). Given any fixed value for the y and x variables, unimodularity and constraint (2.21g) ensure that the w-variables take values of 0 or 1 .

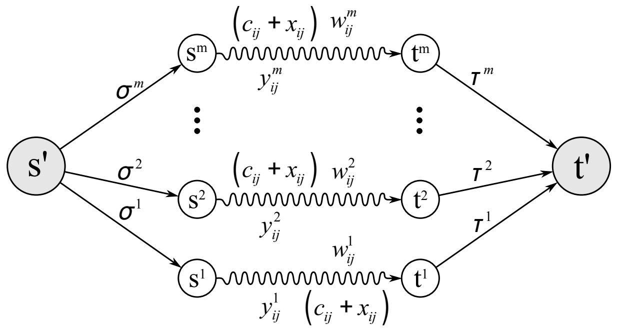


Figure 2.6: **Underlying network model for continuous attack variables** - The original network is stacked m times. We formulate flow conservation constraints from s^k to t^k in terms of y for each layer, which forces the model to select a portfolio of m paths. Arcs are potentially lengthened by the attacker, but lengthened equally for each network layer. To select the shortest remaining path in the portfolio, we formulate another set of flow conservation constraints from s' to t' in terms of σ, w and τ, where flow can only follow a path that is previously selected for the portfolio.

Some of the variables in Model (2.21) minimize while others maximize the objective function value. To overcome that issue, we take the dual of the inner optimization problem twice. Taking the dual of the inner minimization problem in terms of the dual variables defined in brackets yields:

$$
\begin{aligned}
\min_{\boldsymbol{y}} \max_{\boldsymbol{x},\,\boldsymbol{\pi},\,\boldsymbol{\theta},\,\boldsymbol{\rho}} \quad & \theta_s + \theta_t + \sum_{k\in K}\sum_{(i,j)\in A} \rho_{ij}^k y_{ij}^k && && (2.22)\\
\text{s.t.} \quad & -\pi_i^k + \pi_j^k + \rho_{ij}^k \le c_{ij} + x_{ij} && \forall k \in K,\ (i,j)\in A && :\left[w_{ij}^k\right]\\
& \pi_s^k + \theta_s \le 0 && \forall k \in K && :\left[\sigma^k\right]\\
& -\pi_t^k + \theta_t \le 0 && \forall k \in K && :\left[\tau^k\right]\\
& \sum_{(i,j)\in A} x_{ij} \le n && && :[b]\\
& \rho_{ij}^k \le 0 && \forall k \in K,\ (i,j)\in A\\
& \text{(2.21b), (2.21i), (2.21k).}
\end{aligned}
$$

Taking a second dual using dual variables as defined in model (2.22) yields:

$$
\begin{aligned}
\min_{\boldsymbol{y},\,\boldsymbol{w},\,\boldsymbol{\sigma},\,\boldsymbol{\tau},\,b} \quad & \left(\sum_{k\in K}\sum_{(i,j)\in A} c_{ij} w_{ij}^k\right) + n\cdot b && && (2.23a)\\
\text{s.t.} \quad & b \ge \sum_{k\in K} w_{ij}^k && \forall (i,j)\in A \qquad :\left[x_{ij}\right] && (2.23b)\\
& b \ge 0 && && (2.23c)\\
& \text{(2.21b)–(2.21i).}
\end{aligned}
$$

The objective function (2.23a) consists of two competing parts. The first part is the cost of pushing one unit of flow from s' to t' across the paths in the portfolio. To move one unit of flow, the w-variables have to be greater than zero. However, increasing the w-variables requires that we also increase the dual variable b by constraint (2.23b), which forces an increase in the second part of the objective function. To keep the variable b low, the optimization problem splits the flow across the multiple layers of the stacked network, using as many edge disjoint paths as possible. Intuitively, the optimal solution to (2.23) is a portfolio with multiple disjoint paths, such that they have both the same and shortest length after the attack.

It is possible that some paths in the portfolio represented by the y variables contain subtours, because the y's do not show up in the objective function. To reconstruct the optimal, subtour free portfolio, we retrieve the set of disjoint paths from the w-variables. Only arcs (i, j) with $w_{ij}^k > 0$ are in path k of the portfolio. A path with no flow on it ($w_{ij}^k = 0,\ \forall (i, j) \in A$) is not needed, and the portfolio can be decreased in size without affecting the objective function value. To find an optimal attack plan against the portfolio one only needs to know how to distribute the attack budget among the disjoint paths. After the optimal attack, the disjoint paths all have the same length. The difference to the nominal path length tells us how much of the attack budget is put on each path. The exact arc attacked on each path is not relevant because of disjointness.

The C-PPS-m-n optimization model can potentially be used for a commonly occurring application. The optimal solution to the model yields several edge disjoint paths that are both as short as possible and have nearly the same lengths. A common application for this would be providing multiple routes across traffic, routes which are short, use different roads from each other, and all have about the same length. Similar to previous models, C-PPS-m-n can be enhanced with attack probabilities. However, the resulting model does not add new insights in the problem and is therefore left to the reader. The solution for a C-PPS-m-n example on a large network is presented in Section 2.5.5.

2.4 The SNS-m-n Problem

The defender in PPS-m-n moves first and selects a set of m paths, such that the response to the worst-case attack is of minimum length. This first stage is different for the *Sub-Network-Selection-m-n* problem, SNS-m-n. Rather than selecting a portfolio of paths from s to t, the defender extracts a sub-network, consisting of no more than m arcs, from the original network. The remaining stages are similar to PPS-m-n. The attacker destroys up to n arcs in the sub-network, and the defender uses the remaining shortest path in the sub-network for operation.

In SNS-m-n attacked arcs are destroyed, and the defender's final shipping route consists entirely of non-interdicted arcs. Such a route only exists in the sub-network if the attack budget is not sufficient to disconnect source from sink. That means, that $|C|$, the cardinality of the minimum cardinality cutset of the sub-network must be larger than n. We always assume that m is large enough to extract a sub-network with $|C| > n$.

Using similar notation as PPS-m-n, a formulation for SNS-m-n is as follows:

Additional notation

y_{ij}	$y_{ij} = 1$ if arc (i,j) is selected for the sub-network, and $y_{ij} = 0$ otherwise
w_{ij}^E	$w_{ij}^E = 1$ if arc (i,j) is in the shortest path selected by the defender after attack plan E is executed, and $w_{ij}^E = 0$ otherwise

SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ model

$$\min_{\boldsymbol{w},\,\boldsymbol{y},\,v,\,\underline{z}} \quad \underline{z} \tag{2.24a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} y_{ij} - \sum_{i:(j,i)\in A} y_{ji} = \begin{cases} -v & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ v & \text{if } j = t \end{cases} \tag{2.24b}$$

$$\sum_{(i,j)\in A} y_{ij} \leq m \tag{2.24c}$$

$$v \geq n+1 \tag{2.24d}$$

$$\sum_{i:(i,j)\in A} w_{ij}^E - \sum_{i:(j,i)\in A} w_{ji}^E = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall E \in \mathcal{E} \tag{2.24e}$$

$$w_{ij}^E \leq y_{ij} \qquad \forall E \in \mathcal{E},\ (i,j) \in A \tag{2.24f}$$

$$w_{ij}^E = 0 \qquad \forall E \in \mathcal{E},\ (i,j) \in E \tag{2.24g}$$

$$\underline{z} \geq \sum_{(i,j)\in A} c_{ij} w_{ij}^E \qquad \forall E \in \mathcal{E} \tag{2.24h}$$

$$w_{ij}^E \geq 0 \qquad \forall E \in \mathcal{E},\ (i,j) \in A \tag{2.24i}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.24j}$$

With constraints (2.24b)–(2.24d), the defender extracts a sub-network containing no more than m arcs (2.24c), such that the attacker needs to destroy at least $n+1$ of the arcs to disconnect s from t (2.24d). Constraint (2.24b) is the flow conservation constraint for a maximum flow optimization problem, which enforces the desired cutset cardinality because of the duality to the minimum cut problem. Strictly speaking, constraints (2.24b) and (2.24d) are supervalid inequalities, and the formulation would work without them; however, they greatly decrease solution times. For each attack plan, the defender selects a path from s to t (2.24e) that is in the sub-network (2.24f) and is not interdicted (2.24g). The objective function of the problem (2.24a) is the shortest path under the worst-case attack (2.24h).

Each $E \in \mathcal{E}$ generates the set of constraints (2.24e)–(2.24h), and the number of constraints grows as $\binom{|A|}{n}$. This makes the model intractable for larger networks. Similar to PPS-m-n, we use a row-and-column generation approach that sequentially augments an incomplete formulation until the optimal solution is found.

As defined for PPS-m-n, an incomplete model only considers a subset of attack plans, $\overline{\mathcal{E}}$. Let $\boldsymbol{Y}^*$ be the sub-network that is optimal for an incomplete model, with associated objective function value $\underline{z}^*$. In contrast to PPS-m-n, where the solution for incomplete models could be non-operational, here, there does not exist an attack plan that disconnects source from sink in $\boldsymbol{Y}^*$, by constraints (2.24b) and (2.24d). The sub-network $\boldsymbol{Y}^*$ may not be optimal for the complete model, but $\underline{z}^*$ is a valid lower bound because it is the optimal solution of the problem's relaxation.

Because the incomplete model ignores some of the attack plans, one can possibly find a plan $E \notin \overline{\mathcal{E}}$ that forces the defender to respond with a path longer than $\underline{z}^*$. To check the existence for such an attack plan, one needs to solve the SPNI problem with n attacks on $\boldsymbol{Y}^*$. The associated objective function value $\overline{z}^*$ is an upper bound for SNS-m-n, because it represents the objective function value of SNS-m-n for the specific feasible solution $\boldsymbol{Y}^*$. If $\underline{z}^* = \overline{z}^*$, then $\boldsymbol{Y}^*$ is optimal for the complete model.

A sequence for the row-and-column generation procedure is depicted in Figure 2.7. In each iteration, solving the incomplete model provides the lower bound. The best upper bound can be maintained from the feasible solutions seen thus far.

Section 2.5.6 contains two examples for SNS-m-n. For a small network, we solve SNS-m-n with different parameters m and show how the size of the sub-network impacts the optimal solution. For a large network, we draw the optimal solution for SNS-m-n on a map and plot the converging bounds that are yielded by the row-and-column generation procedure.

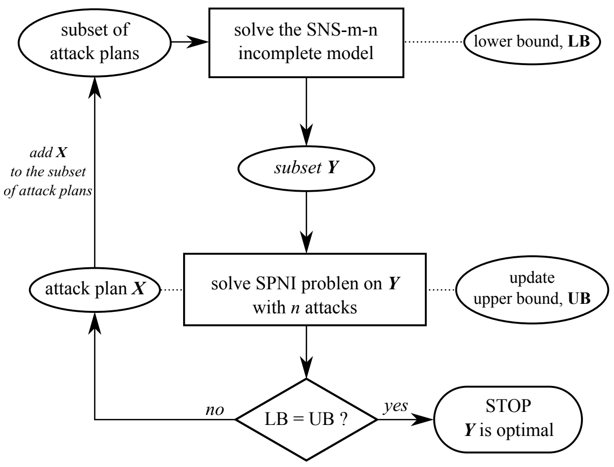


Figure 2.7: **Row-and-column generation procedure for SNS-m-n** - We start with a subset of attack plans and find the optimal solution Y for the resulting incomplete model. Such a solution provides a lower bound for the length of the response path, and is a feasible sub-network for the complete model. Solving SPNI on the sub-network yields an upper bound. As long as lower and upper bounds do not agree, Y is suboptimal, and we add the worst-case attack plan against Y to the subset of attack plans.

2.4.1 Pr-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ with Attack Probabilities

Similar to previous problems, we can incorporate attack probabilities in model (2.24). To do so, we must consider attack plans $E_\ell \in \mathcal{E}_n$ of cardinality $|E_\ell| \in \{0, \dots, n\}$ as follows:

Pr-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ model

$$\min_{\boldsymbol{w},\, \boldsymbol{y},\, \underline{\boldsymbol{z}},\, v} \quad \sum_{\ell=0}^{n} p_\ell \, \underline{z}_\ell \tag{2.25a}$$

$$\text{s.t.} \quad \sum_{i:(i,j)\in A} y_{ij} - \sum_{i:(j,i)\in A} y_{ji} = \begin{cases} -v & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ v & \text{if } j = t \end{cases} \tag{2.25b}$$

$$\sum_{(i,j)\in A} y_{ij} \le m \tag{2.25c}$$

$$v \ge n + 1 \tag{2.25d}$$

$$\sum_{i:(i,j)\in A} w_{ij}^{E_\ell} - \sum_{i:(j,i)\in A} w_{ji}^{E_\ell} = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N\backslash\{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall E_\ell \in \mathcal{E}_n \tag{2.25e}$$

$$w_{ij}^{E_\ell} \le y_{ij} \qquad \forall E_\ell \in \mathcal{E}_n,\ (i,j) \in A \tag{2.25f}$$

$$w_{ij}^{E_\ell} = 0 \qquad \forall E_\ell \in \mathcal{E}_n,\ (i,j) \in E_\ell \tag{2.25g}$$

$$\underline{z}_\ell \ge \sum_{(i,j)\in A} c_{ij} w_{ij}^{E_\ell} \qquad \forall \ell = 0, \dots, n,\ E_\ell \in \mathcal{E}_n \tag{2.25h}$$

$$w_{ij}^{E_\ell} \ge 0 \qquad \forall E_\ell \in \mathcal{E}_n,\ (i,j) \in A \tag{2.25i}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.25j}$$

In the worst case, the defender's sub-network must withstand an attack on n arcs, and still provide a path from s to t. Therefore, constraints (2.25b)–(2.25d) remain unchanged in comparison to model (2.24). However, the path that remains shortest after an attack on ℓ arcs depends on ℓ. As a consequence, constraints (2.25e)–(2.25i) distinguish attacks according to the number of attacked arcs ℓ. Constraint (2.25h) computes the length of the defender's response to an attack on ℓ arcs. These responses are weighted with the attack probabilities in the objective function (2.25a) to yield the expected length of the defender's response.

Following the row-and-column generation approach, as shown in Figure 2.7, one here has to solve the SPNI problem for each ℓ and compute the resulting upper bound. Each SPNI problem returns an optimal attack plan E_ℓ, for $\ell = 1, \dots, n$, that needs to be added to the incomplete model until the optimal solution is found.

2.4.2 D-SNS-m-n with General Attacks

Similar to D-PPS-m-n, an attacked arc (i, j) in D-SNS-m-n is not destroyed, but lengthened by a predefined length d_{ij}. Under these circumstances, it may be optimal to use an attacked path as response to the worst-case attack. Therefore, the model does not contain requirements for the sub-network's minimum cutset, and is formulated as follows:

D-SNS-m-n model

$$\min_{\boldsymbol{w}, \boldsymbol{y}, \underline{z}} \quad \underline{z} \tag{2.26a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} y_{ij} \le m \tag{2.26b}$$

$$\sum_{i:(i,j)\in A} w^E_{ij} - \sum_{i:(j,i)\in A} w^E_{ji} = \begin{cases} -1 & \text{if } j = s \\ 0 & \text{if } j \in N \backslash \{s,t\} \\ 1 & \text{if } j = t \end{cases} \qquad \forall E \in \mathcal{E} \tag{2.26c}$$

$$w^E_{ij} \le y_{ij} \qquad \forall E \in \mathcal{E},\ (i,j) \in A \tag{2.26d}$$

$$\underline{z} \ge \sum_{(i,j)\in A} c_{ij} w^E_{ij} + \sum_{(i,j)\in E} d_{ij} w^E_{ij} \qquad \forall E \in \mathcal{E} \tag{2.26e}$$

$$w^E_{ij} \ge 0 \qquad \forall E \in \mathcal{E},\ (i,j) \in A \tag{2.26f}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.26g}$$

Constraint (2.26b) is a budget constraint that restricts the defender to extract a sub-network with no more than m arcs. From the sub-network, we select a s-t-path by constraints (2.26c) and (2.26d), and compute its length under the worst-case attack by (2.26e). The length under the worst-case attack serves as objective function (2.26a). Minimizing the objective function enforces a sub-network with corresponding s-t-path of minimum length under the worst-case attack.

2.4.3 C-SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ with Continuous Attack Variables

Similar to C-PPS-m-n, we can define the *Continuous-Sub-Network-Selection-m-n* problem, C-SNS-m-n. The attacker in C-SNS-m-n distributes the attack budget, representing a total length that can be added to the network, continuously across the arcs in the defenders' extracted sub-network. We introduce new variables and formulate the problem as follows:

Additional notation

w_{ij}	proportion of flow from s to t that is pushed across arc (i, j)
x_{ij}	length added to the nominal length c_{ij} if arc (i, j) is attacked

C-SNSmnbold model, primal formulation

$$\min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \min_{\boldsymbol{w}} \sum_{(i,j)\in A} w_{ij}(c_{ij} + x_{ij}) \tag{2.27a}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} w_{ij} - \sum_{i:(j,i)\in A} w_{ji} = \begin{cases} -1 & \text{if } j = s & : [\pi_s] \\ 0 & \text{if } j \in N\backslash\{s,t\} & : [\pi_j] \\ 1 & \text{if } j = t & : [\pi_t] \end{cases} \tag{2.27b}$$

$$w_{ij} \leq y_{ij} \qquad \forall (i,j) \in A \quad : \left[\rho_{ij}\right] \tag{2.27c}$$

$$\sum_{(i,j)\in A} y_{ij} \leq m \tag{2.27d}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.27e}$$

$$w_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{2.27f}$$

$$\sum_{(i,j)\in A} x_{ij} \leq n \tag{2.27g}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{2.27h}$$

The flow conservation constraint (2.27b) ensures the defender selects a path from s to t, exclusively consisting of arcs from the extracted sub-network by (2.27c). The size of the sub-network is restricted by constraint (2.27d). The length of the defender's response path serves as the objective function (2.27a) and is potentially lengthened by the attacker, who is restricted by the budget constraint (2.27g). We obtain a single minimization problem by taking the dual of the inner optimization problem twice, as follows:

$$
\begin{aligned}
\min_{\boldsymbol{y}} \max_{\boldsymbol{x},\,\boldsymbol{\pi},\,\boldsymbol{\rho}} \quad & \pi_t - \pi_s + \sum_{(i,j)\in A} \rho_{ij} y_{ij} && && \text{(2.28a)}\\
\text{s.t.} \quad & \pi_j - \pi_i + \rho_{ij} \le c_{ij} + x_{ij} && \forall (i,j) \in A \quad : [w_{ij}] && \text{(2.28b)}\\
& \sum_{(i,j)\in A} x_{ij} \le n && \quad : [b] && \text{(2.28c)}\\
& \rho_{ij} \le 0 && \forall (i,j) \in A && \text{(2.28d)}\\
& \text{(2.27d), (2.27e), (2.27h)}
\end{aligned}
$$

and taking a second dual using dual variables as defined in model 2.28 yields:

$$
\begin{aligned}
\min_{\boldsymbol{y},\,\boldsymbol{w},\,b} \quad & \left(\sum_{(i,j)\in A} c_{ij} w_{ij} \right) + n \cdot b && && \text{(2.29a)}\\
\text{s.t.} \quad & b \ge w_{ij} && \forall (i,j) \in A \quad : [x_{ij}] && \text{(2.29b)}\\
& b \ge 0 && && \text{(2.29c)}\\
& \text{(2.27b)–(2.27f)}
\end{aligned}
$$

The objective function (2.29a) consists of two competing parts. The first part is the cost of pushing one unit of flow from s to t in the sub-network, enforced by the flow conservation constraint (2.27b). To move one unit of flow, the w-variables have to be greater than zero. However, increasing the w-variables requires an increase of the dual variable b by constraint (2.29b), which forces an increase in the second part of the objective function. Intuitively, the optimal solution to (2.29) is a sub-network with multiple disjoint paths, such that they have both the same and shortest length after the attack.

An example for C-SNS-m-n on a large network is shown in Section 2.5.7.

2.4.4 Generalization of SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ to Arbitrary Network Problems

In our work, we primarily concentrate on models that minimize the length of the shortest path in the post-attack network. In particular, the objective in SNS-m-n, model (2.24), is the length of the defender's response path and constraints (2.24e)–(2.24f) ensure the defender selects a path after the attack. We can replace the objective and constraints with an arbitrary, general minimum cost flow problem. The resulting model is

Additional notation

b_j	supply or demand for node j, with $b_j < 0$ being supply and $b_j > 0$ being demand that needs to be cleared
u_{ij}	maximum flow that can be put on arc (i, j); also called arc capacity
c_{ij}	cost one has to pay for putting one unit of flow on arc (i, j)

SNS-$\boldsymbol{m}$-$\boldsymbol{n}$ model with minimum cost flow objective

$$\begin{aligned}
\min_{\boldsymbol{w},\, \boldsymbol{y},\, v,\, \underline{z}} \quad & \underline{z} && && (2.30a)\\
\text{s.t.} \quad & \sum_{(i,j)\in A} y_{ij} \le m && && (2.30b)\\
& \sum_{i:(i,j)\in A} w_{ij}^E - \sum_{i:(j,i)\in A} w_{ji}^E = b_j && \forall j \in N,\ E \in \mathcal{E} && (2.30c)\\
& w_{ij}^E \le y_{ij} u_{ij} && \forall E \in \mathcal{E},\ (i,j) \in A && (2.30d)\\
& w_{ij}^E = 0 && \forall E \in \mathcal{E},\ (i,j) \in E && (2.30e)\\
& \underline{z} \ge \sum_{(i,j)\in A} c_{ij} w_{ij}^E && \forall E \in \mathcal{E} && (2.30f)\\
& w_{ij}^E \ge 0 && \forall E \in \mathcal{E},\ (i,j) \in A && (2.30g)\\
& y_{ij} \in \{0,1\} && \forall (i,j) \in A && (2.30h)
\end{aligned}$$

The sub-network containing no more than m arcs from the original network is selected in model (2.30) by constraint (2.30b). Constraints (2.30d) and (2.30e) ensure that only non-interdicted arcs in the sub-network carry flow, bounded by the arc's capacity. A feasible flow satisfies the network's demand and supply by constraint (2.30c), incurring a cost for using the arcs in the network. The worst-case attack induces the highest cost, which is selected by constraint (2.30f) and serves as objective function of the minimization problem.

Model (2.30) is a generalization of SNS-m-n to other network optimization problems, and can be solved with the same row-and-column generation approach we use for SNS-m-n. However, significant additional work may be required to make the approach practicable. For example, the supervalid inequalities (2.24b) and (2.24d) significantly reduce runtimes for SNS-m-n. Similar supervalid inequalities could help in the more general minimum cost flow formulation. In addition, network simplification steps as the ones in section 2.3.5 could reduce problem size and produce solutions for large networks. The generalization of SNS-m-n to minimum cost flow problems significantly expands the applicability of the models we introduce in this paper.

2.5 Examples and Computational Results

2.5.1 Influence of Portfolio Size on the Optimal Solution

The following example depicts the general idea of the Path-Portfolio-Selection problem and supports Lemma 3. For the "six-hop-zigzag" graph in Figure 2.8, we solve PPS-m-1 for different portfolio sizes, $m \in \{2, 3, 6\}$. The optimal solutions are depicted in Figure 2.9. For each portfolio size, we get a different portfolio as optimal solution. As expected, the responses by the defender to the optimal attack have shorter lengths for larger portfolio sizes. With a portfolio size of six paths, the defender is able to achieve the SPNI solution, and there is no benefit gained by increasing the portfolio size beyond six paths.

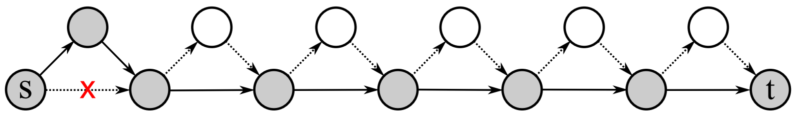


Figure 2.8: **Six-hop zigzag graph** - The graph consists of 13 nodes and 18 arcs. Each of the arcs has a length of one. Source and sink are labeled accordingly. The solution to the SPNI problem is also represented. One of the attacker's optimal attack plans interdicts the arc which is crossed out in red. The defender's response to this attack is the path including the seven solid arcs and grayed nodes, with a total length of seven.

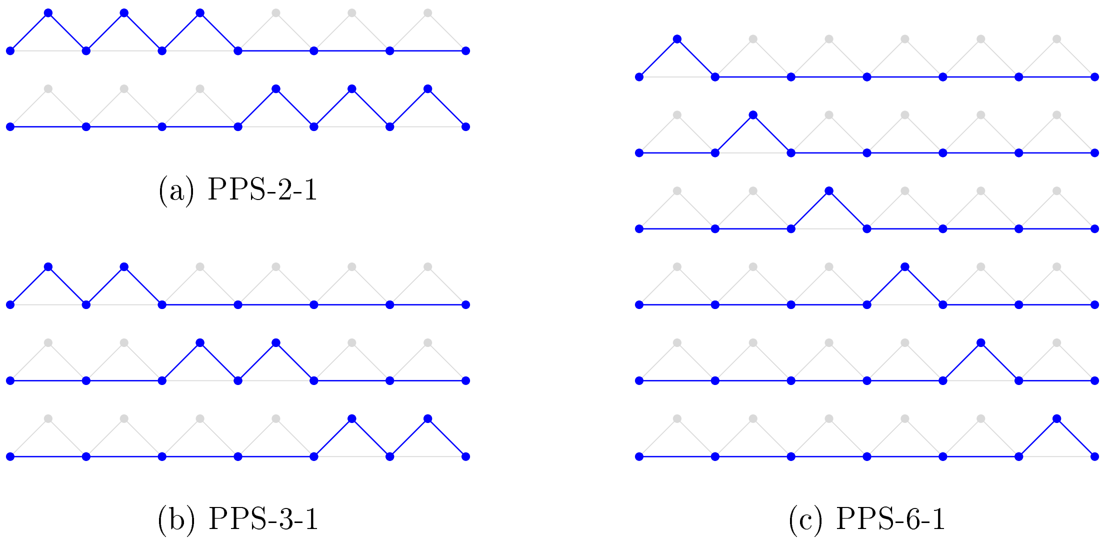


Figure 2.9: **Six-hop zigzag PPS-*m*-1** - (a) PPS-2-1 is an instance of PPS-m-$(m-1)$, and the optimal portfolio necessarily contains two disjoint paths. The attacker is restricted to attack one of the paths. Each path has a length of nine units, hence the defender's response is a path of length nine. (b) The paths in the defender's portfolio are not disjoint, but no arc appears in all of the paths. The worst-case attack interdicts two of the paths. The length of the longest path in the portfolio is eight, as compared to nine in PPS-2-1. A further increase of the portfolio size up to five has no effect on the length of the longest path. (c) The portfolio contains six paths, each of length seven, which is the value of the SPNI, and therefore a lower bound. Adding more paths to the portfolio does not affect the defender's response, by Lemma 3.

2.5.2 Two Examples for PPS-5-2 on a Large Network

We create two case studies with large networks and demonstrate the impact of sequentially bounding and shrinking the network. The data is extracted from the Los Alamos Pathway Analysis, Threat Response and Interdiction Options Tool (PATRIOT).

The first case study is a road network of Germany, consisting of 2,971 nodes and 8,824 arcs. We select Hamburg as the source and Munich as the sink. The shortest paths between source and sink consists of 76 arcs with a total length of 677,898 meters. The shortest path under the worst attack destroying two arcs, where the defender can use the entire network, has a length of 695,773 meters. The Germany network, the shortest path, and the path taken after two worst-case attacks are shown in Figure 2.10.

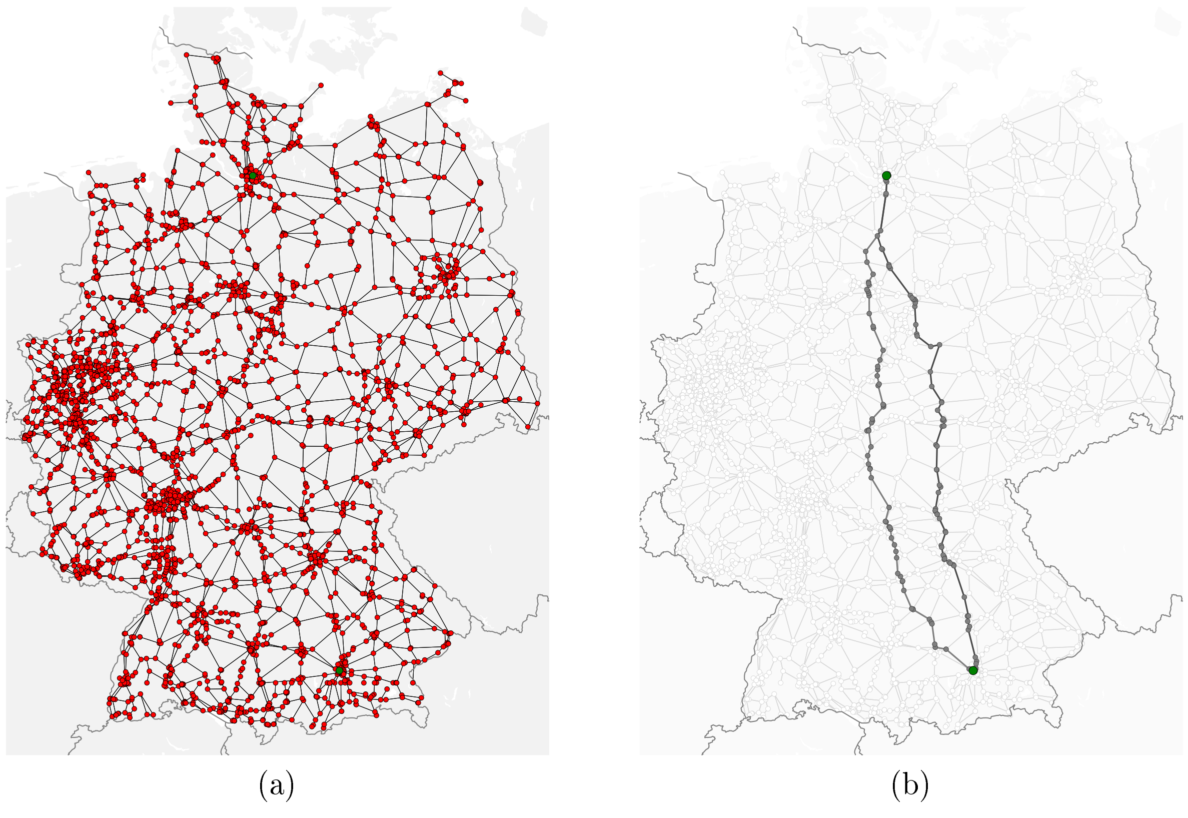


Figure 2.10: **Germany road network** - (a) The network with 2,971 nodes and 8,824 arcs. (b) The shortest path between Hamburg and Munich with a total length of 677,898 meters (left path) and the solution to the SPNI problem with a length of 695,773 meters (right path).

Solving the complete model (2.3) for PPS-5-2 is impossible. There are 194,679,501 variables and 817,512,726 constraints in the complete model. The required memory exceeds the available 8 GB in our computer. The constraint generation approach without supervalid inequalities is not practicable either. We stopped the solution process after 36 hours without a feasible solution in hand.

We follow the approach from Section 2.3.5 and find bounds that are used to shrink the network. Figures 2.11–2.13 show different feasible solutions and the resulting network after the shrinking process. Finally, we solve PPS-5-2 on the shrunken network using supervalid inequalities and the constraint generation approach. The attack plans found while solving the SPNI problem are used as an initial set of attack plans. The associated runtimes are shown in Table 2.1.

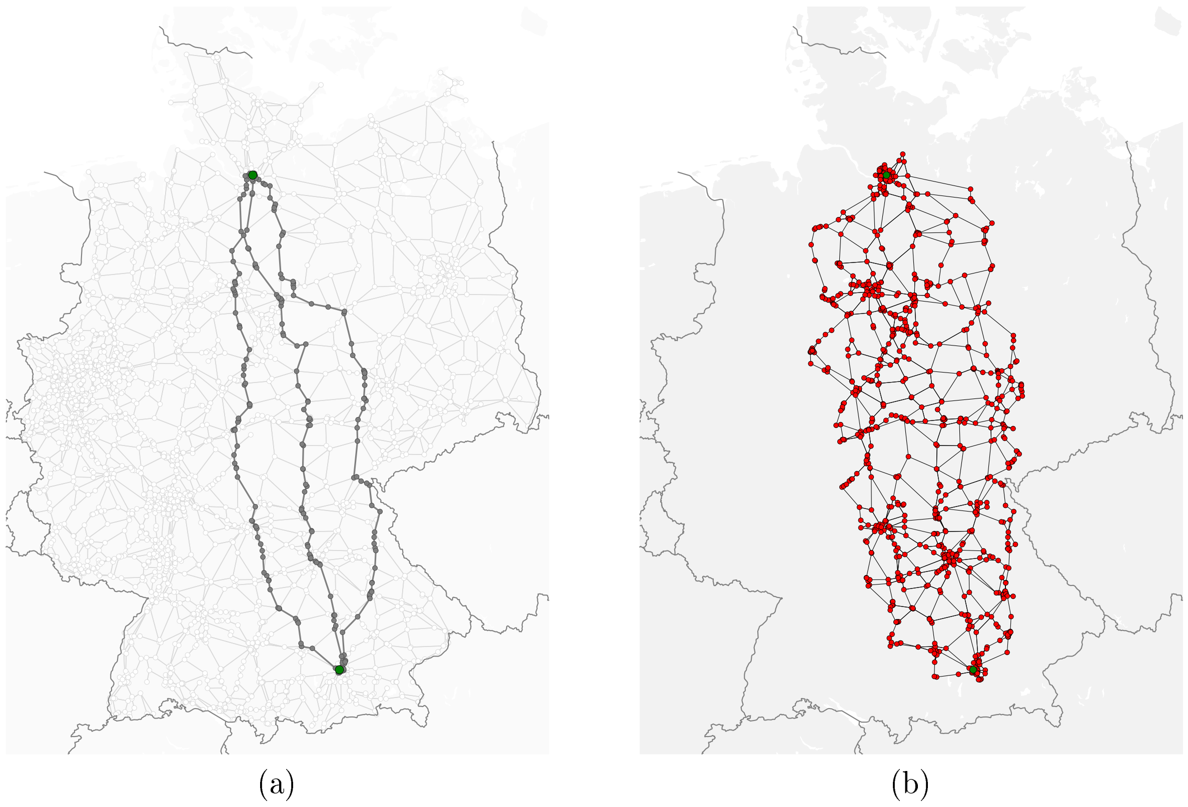

(a) (b)

Figure 2.11: **Sequentially disjoint paths, Germany** - (a) The three sequentially found disjoint paths form an operational portfolio, providing an upper bound of 739,092 meters. (b) The network after removing nodes and arcs according to this upper bound, with 813 nodes and 2,213 arcs.

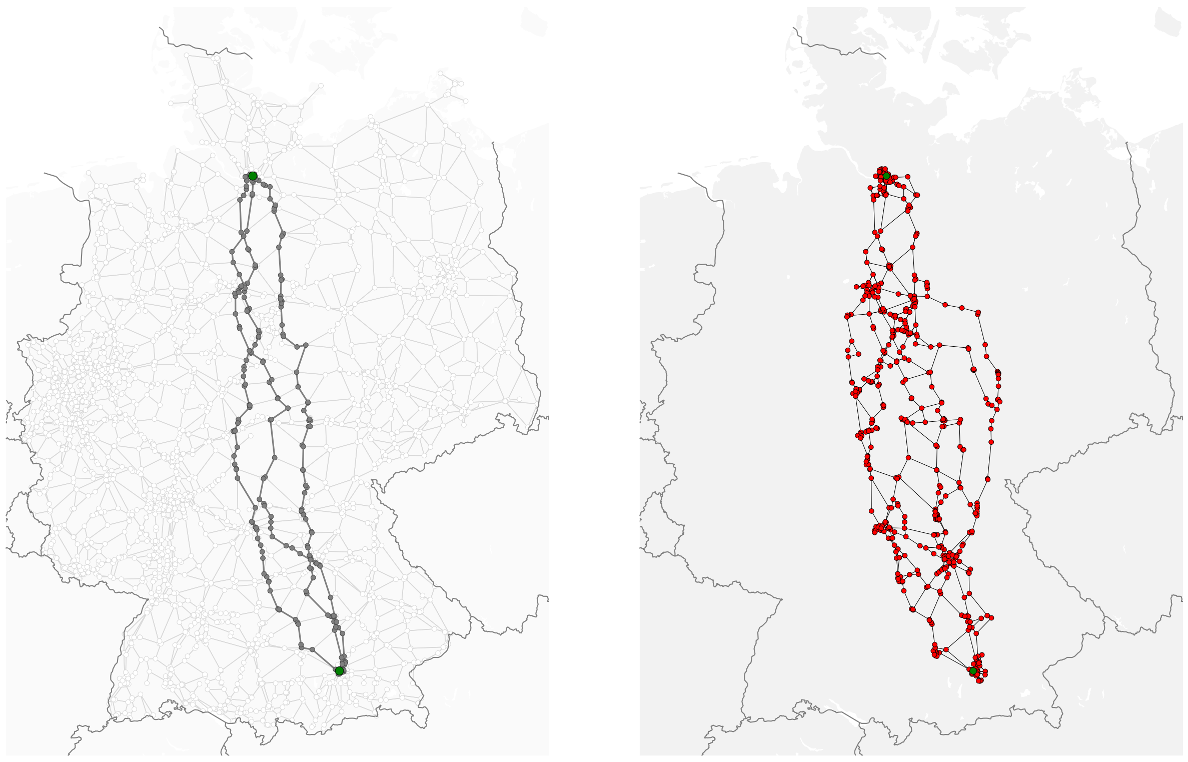
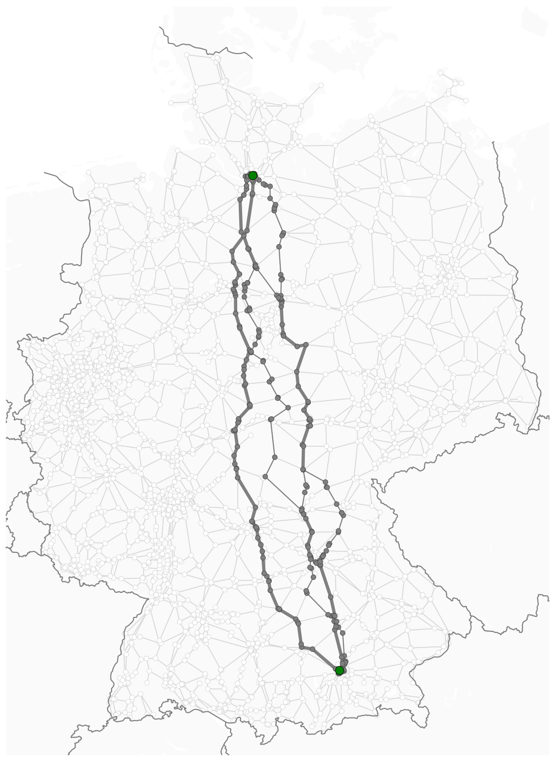

(a) (b)

Figure 2.12: **Optimized disjoint portfolio, Germany** - (a) Three disjoint paths such that the length of the longest path is minimized. The three paths form an operational portfolio and provide an upper bound of 707,713 meters. (b) The network after removing nodes and arcs according to this upper bound, with 523 nodes and 1,297 arcs.


(a)

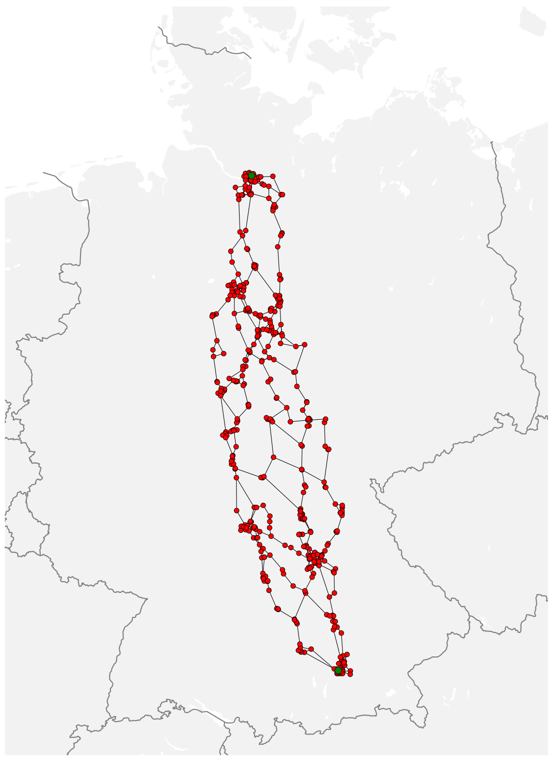
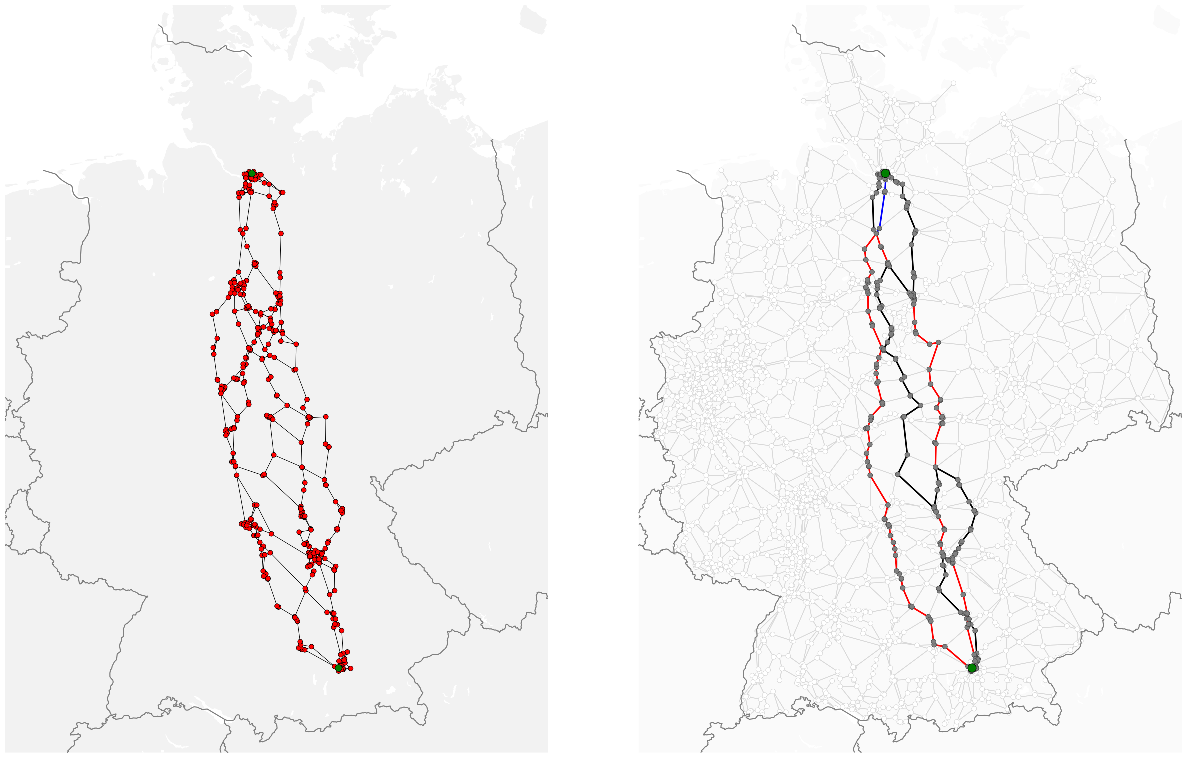

(b)

Figure 2.13: **Optimized portfolio with arc use constraints, Germany** - (a) Five paths in the portfolio, such that the longest is of minimum length, and each arc does not appear in more than two paths. The thicker arcs are part of two different paths. The upper bound produced by this solution is 700,947 meters. (b) The network after removing nodes and arcs according to this upper bound, with 447 nodes and 1,078 arcs.



(a) (b)

Figure 2.14: **Optimal solution, Germany** - (a) The network after the final simplifications with 367 nodes and 908 arcs. (b) The optimal solution for the PPS-5-2 problem. The red arcs are part of two different paths, and the blue arcs are included in three different paths. Using an arc in three different paths is infeasible for the model that finds the best upper bound (2.15). The longest path in the portfolio has a length of 700,668 meters.

Table 2.1 summarizes the results from the plots in Figures 2.10–2.14, including the runtimes to solve the models. While 36 hours were not enough time to find any solution using a direct constraint generation approach, a cumulative runtime of less than 2.5 hours was sufficient to find the optimal solution using supervalid inequalities and network size reductions. Notice the remarkably small optimality gaps for the intermediate steps. The solution process finds a solution with an optimality gap of less than one percent within less than ten minutes. Practically, one could stop the solution procedure at this point. Except for the last row in Table 2.1 the optimality gap is calculated as

$$\frac{UB - LB}{LB} \cdot 100\%$$

and the final optimality gap results from the MIP solver settings.

model	lower bound	upper bound	network size	runtime	optimality gap
(2.5)	695,773		2,971/8,824	0:00:13	
seq. disj.		739,092	2,971/8,824	0:00:01	6.22%
(2.14)		707,713	813/2,213	0:00:24	1.72%
(2.15)		700,947	523/1,297	0:06:15	0.74%
(2.3)		700,668	367/908	2:19:42	0.20%

Table 2.1: **PPS-5-2 results for the German network** - Model complexity and runtimes increase in lower rows of the table. The optimality gaps decrease in lower rows. The upper bounds from simpler models are used to shrink the network for more complex models. The size of the network is reported as a tuple of number of nodes and number of arcs. Model (2.5) solves the SPNI problem and provides the lower bound. The first upper bound comes from a set of sequentially disjoint paths. Model (2.14) seeks a set of disjoint paths, such that the longest is minimized. In model (2.15), we restrict each arc from appearing too many times. Model (2.3) solves PPS-5-2 using both supervalid inequalities and the constraint generation approach. Model (2.15) provides a remarkably small optimality gap with a short solution time.

Second case study, a U.S. road network

As a second case study, we bound the PPS-5-2 problem for a U.S. road network, which consists of 47,941 nodes and 124,414 arcs. We use Washington, D.C., as the source and Los Angeles as the sink. The shortest path between the two cities contains 453 arcs with a total length of 4,065,926 meters. The solution to the SPNI problem with two attacks is a path of length 4,095,224 meters. The data for the solution procedure is shown in Table 2.2. Unlike the example for Germany, we do not solve model (2.3). The formulation quickly exceeds the available memory of 8 MB, even after shrinking and simplifying the network as much as possible. In order to cut runtime we solve (2.15) two times. The first time we set the MIP optimality gap to a relatively large value, and use the solution to shrink the network one more time. The optimality gap of less than 0.1% resulting from these steps makes the optimal solution to model (2.3) practically unnecessary.

model	lower bound	upper bound	network size	runtime	optimality gap
(2.5)	4,095,224		47,941/124,414	0:21:14	
seq. disj.		4,216,288	47,941/124,414	0:02:18	2.90%
(2.14)		4,113,892	7,367/16,014	0:17:14	0.45%
(2.15)		4,101,747	5,536/11,496	0:06:23	0.16%
(2.15)		4,098,163	4,987/10,244	0:14:00	0.07%

Table 2.2: **PPS-5-2 results for the U.S. network** - Model complexity and runtimes increase in lower rows of the table. The optimality gaps decrease in lower rows. The upper bounds from simpler models are used to shrink the network for more complex models. The size of the network is reported as a tuple of the number of nodes and number of arcs. Model (2.5) solves the SPNI problem and provides the lower bound. The first upper bound comes from a set of sequentially disjoint paths. Model (2.14) seeks a set of disjoint paths, such that the longest is minimized. In model (2.15), we restrict each arc from appearing too many times. Here we solve (2.15) two times with different solver settings, and shrink the network one more time. Again, model (2.15) provides a remarkably small optimality gap with a short solution time.

2.5.3 An Example for Pr-PPS-m-n

The following example serves two purposes. First, it demonstrates how attack probabilities can influence the optimal portfolio. Second, it shows that model (2.19) bounds Pr-PPS-m-n but does not necessarily provide the optimal solution, which can be found by solving the more complex model (2.16). We use the four-hop zigzag graph with a shortcut in Figure 2.15 as example network.

The optimal portfolio for PPS-4-2 is shown in Figure 2.16 (a). PPS-4-2 does not consider attack probabilities, and one can find the same solution by solving model (2.16) with attack probability $p_2 = 1$. Now, we define the attack probabilities as $p_2 = 0.25$, $p_1 = 0.75$ and $p_0 = 0$, but retain the optimal portfolio from PPS-4-2. The expected length of the defender's response path is 5.25. Incorporating the attack probabilities and solving model (2.19) quickly returns a feasible portfolio with a performance guarantee for Pr-PPS-4-2 shown in Figure 2.16 (b). The performance guarantee bound for using this portfolio, resulting from model (2.19), is an expected response length of 5.0. The bound and the true performance for this portfolio agree. The optimal portfolio for Pr-PPS-4-2 is shown in Figure 2.16 (c), has an expected response length of 4.5, and is found by solving model (2.16).

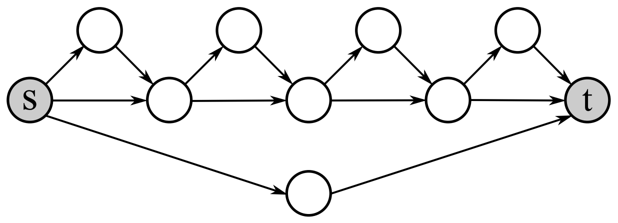


Figure 2.15: **Four-hop zigzag with shortcuts** - The graph contains 10 nodes and 15 arcs. Each arc has a length of 1 unit. We solve PPS-4-2 with and without incorporating attack probabilities. The solution for the SPNI problem with one attack has a length of four units, and a length of five units in case of two attacks. A lower bound for the Pr-PPS-4-2 with the defined probabilities, $p_2 = 0.25$ and $p_1 = 0.75$, is therefore 4.25.

To emphasize the impact of attack probabilities, Figure 2.17 shows the expected length of the response as a function of p_1 for the three portfolios from Figure 2.16. The plot depicts that depending on the attack probabilities, different portfolios are optimal. Here, one can find a dominant portfolio that is optimal no matter the attack probabilities. Certainly, such a portfolio does not always exist.

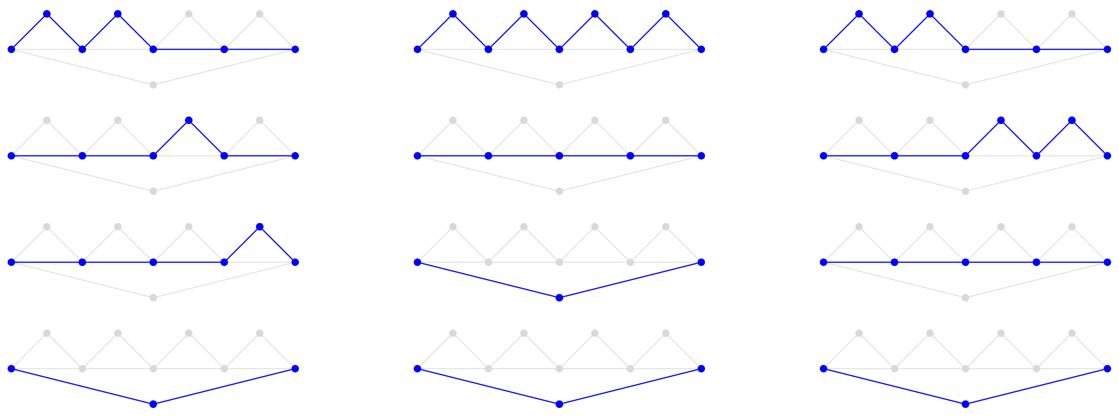


Figure 2.16: **Four-hop zigzag with shortcuts, PPS-4-2** - (a) The figure shows the optimal solution for PPS-4-2. If we use this portfolio for the Pr-PPS-4-2 problem, with probability $p_1 = 0.75$, the attacker interdicts one arc only and the defender responds with a path of length five; with probability $p_1 = 0.25$ the attacker interdicts two arcs, and the defender responds with a path of length six. Expected length of the defender's response is 5.25. (b) To bound Pr-PPS-4-2 we assume that the path indexed by $k = 3$ is the response to the optimal attack on one arc, and compute a portfolio using model 2.19. The figure shows the optimal portfolio, which has an objective function value of 5. For this portfolio, the objective function value of model 2.19 happens to agree with the true expected response, but this is not always true. However, the objective function value of model 2.19 always gives an upper bound for Pr-PPS-4-2. (c) The optimal solution for Pr-PPS-4-2, which outperforms the other two portfolios, and is found by solving model (2.16).

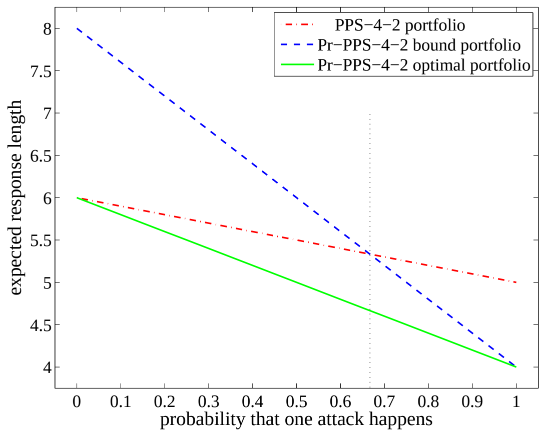


Figure 2.17: **Four-hop zigzag with shortcuts, changing probabilities** - The plot shows the expected response length for the portfolios from Figure 2.16. The optimal solution for Pr-PPS-4-2 has the smallest expected length regardless of the attack probability. Assume we do not have the solution for Pr-PPS-4-2 because it is intractable to solve. As long as the attack probability $p_1 \leq \frac{2}{3}$ the defender prefers the optimal portfolio for PPS-4-2, and selects the portfolio that bounds Pr-PPS-4-2 if $p_1 > \frac{2}{3}$.

To test model (2.19) on a large network, which returns a portfolio with a performance guarantee quickly, we use the German network with Hamburg as source and Offenbach as sink. The shortest path between source and sink consists of 55 arcs, with a total length of 442,916 meters. We define the probabilities: $p_0 = 0$, $p_1 = 0.7$ and $p_2 = 0.3$. The shortest path in the SPNI problem with one attack has a length of 451,237 meters and for two attacks, it has a length of 462,995 meters. This gives a lower bound of 454,765 meters for the expected defender response. We use that lower bound to compute the optimality gap in Table 2.3. Solving PPS-5-2 under certainty, with $p_2 = 1.0$, gives a particular feasible portfolio and an upper bound to the Pr-PPS-5-2 problem. Specifically, we can attack the portfolio resulting from PPS-5-2 with a single attack and find the defender's response. With that response we can compute the upper bound for Pr-PPS-5-2.

We sequentially bound the length of the longest path in the optimal portfolio and shrink the network accordingly, as described in Sections 2.3.5 and 2.3.6. After shrinking the network as much as possible, we solve model 2.19 to bound Pr-PPS-5-2 in 3:53:13 hours. Table 2.3 shows the path length for the different portfolios that are optimal to the bounding models.

	PPS-m-n model	disjoint solution	arc use limit	bounding model
$l(k = 5)$	462,995	467,595	465,652	474,785
$l(k = 4)$	462,988	465,521	463,081	455,535
$l(k = 3)$	462,799	463,585	463,068	455,278
$l(k = 2)$	461,773		462,995	455,266
$l(k = 1)$	461,521		461,876	454,321
expectation	462,990	466,143	463,853	461,310
gap [%]	1.81	2.50	1.99	1.44

Table 2.3: **Pr-PPS-5-2 results for the German network** - Each row in the table represents one path in a portfolio, and each column represents a portfolio derived using a different model. The first column represents the portfolio from solving PPS-5-2. This solution serves as reference and is not part of the bounding and shrinking procedure, because of its large solution time. The second column holds the lengths of the optimal disjoint portfolio from model (2.14), and the third column holds the optimal portfolio with arc use constraints as described for constraints (2.15). From each portfolio, we can compute an upper bound for Pr-PPS-5-2 and shrink the network accordingly. The network was shrunk to a size of 555 nodes and 1,690 arcs. The fourth column represents solving model (2.19), producing our best feasible solution for Pr-PPS-5-2 in 3:53 hours. We do not solve the Pr-PPS-5-2 to optimality using model (2.16).

2.5.4 An Example for D-PPS-m-n

We create the graph as shown in Figure 2.18 and solve different instances of PPS-m-n and D-PPS-m-n. With delays $d_{ij} \forall (i,j) \in A$ that are large enough to represent infinity, PPS-m-n and D-PPS-m-n are the same problem and the optimal solutions match. In this way, we can compare runtimes and check if the model results agree. We solve PPS-m-n using the constraint generation approach, without incorporating supervalid inequalities. The result shows that PPS-m-n solves much faster, and D-PPS-m-n quickly reaches the limits of tractability. Runtimes and objective function values are shown in Table 2.4.

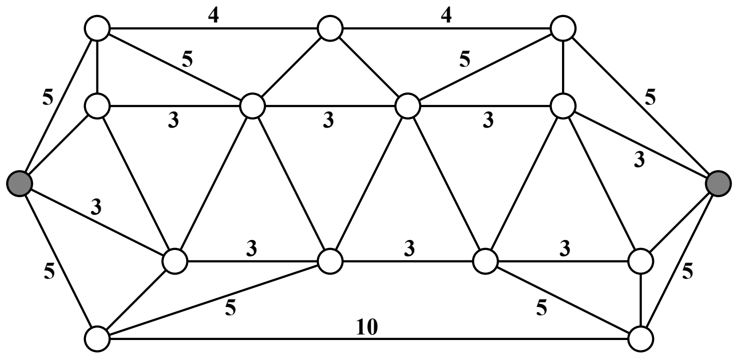


Figure 2.18: **Example network** - The example network consists of 15 nodes and 64 arcs. It is drawn as undirected network, because there is an arc in each direction of same length. The label represents the associated arc length. Unlabeled arcs have a length of one. Source and sink are highlighted in gray.

model	response length	number iterations	runtime [sec]
PPS-3-2	14	1	4
D-PPS-3-2	14	17	22
PPS-4-2	13	3	6
D-PPS-4-2	13	28	2,270

Table 2.4: **Results for the D-PPS-m-n example** - The objective function value for PPS-m-n and D-PPS-m-n match because we use delays $d_{ij} = M$, but PPS-m-n outperforms D-PPS-m-n in runtime. Remarkable is the significant increase in runtime from D-PPS-3-2 to D-PPS-4-2, compared to the moderate increase in iterations for the row-and-column generation procedure. This indicates that the incomplete models for D-PPS-4-2 are difficult to solve, and likely have poor LP relaxations. The increase in runtime indicates that the D-PPS-m-n model is probably intractable for bigger networks without additional constraints or algorithmic work. Because we use the set of attack plans found while solving SPNI as initial attack set for PPS-m-n, the number of iterations of the row-and-column generation procedure here is low. A portfolio of size 4 is sufficiently large to yield the lower bound of 13 for the defender's response length. Therefore, an increase of m beyond 4 is not effective.

As a second example for D-PPS-m-n, we resort to the six-hop zigzag graph and solve D-PPS-3-2 with delays $d_{ij} < M$. The randomly chosen delays are shown in Figure 2.19. The optimal portfolio is shown in Figure 2.20.

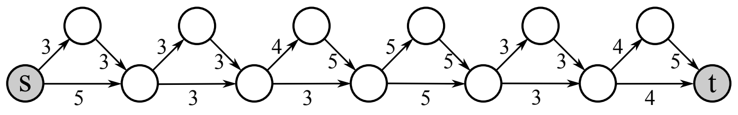


Figure 2.19: **Six-hop zigzag with delays** - The nominal length of each arc in the network is one. The label in the figure represent the delay d_{ij} for each arc, which was randomly assigned. For the shown network we solve D-PPS-3-2.

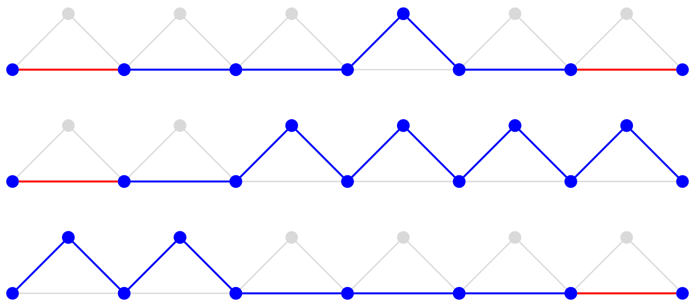


Figure 2.20: **Optimal solution for D-PPS-3-2** - The portfolios' path lengths are 7, 10, and 8. The attacked arcs are highlighted in red. The remaining shortest path in the attacked portfolio is the bottom one, with a length of length 12.

2.5.5 An Example for C-PPS-m-n

For this example we use the German network from Figure 2.10 with its 2,971 nodes and 8,824 arcs and assume Hamburg as source and Munich as sink. The shortest path between the two nodes contains 76 arcs with a total length of 677,898 meters. The portfolio size is set to 5 and the attacker's budget suffices to cause a delay that is equivalent to adding a total length of 200,000 meters to the network. The optimal solution contains three disjoint paths. The three paths' lengths and the budget the attacker assigned to each path are shown in Table 2.5. The two additional paths have no impact to the solution, the associated w-variables are zero. The C-PPS-5-200000 problem solves in 1:47 minutes on a modern computer.

disjoint path	path length	attack value on path	length after attack
1	694,591	75,231.66	769,822.66
2	703,701	66,121.66	769,822.66
3	711,176	58,646.66	769,822.66

Table 2.5: **Results for the C-PPS-5-200000 example** - We use the Germany network with its 2,971 nodes and 8,824 arcs. Hamburg is the assumed source and Munich serves a s sink. The optimal solution contains three disjoint paths with length as shown in the table. After the attack, the length of these three paths is equal and the shortest length in the defender's portfolio. The defender arbitrarily picks one of the paths as response to the attack.

2.5.6 An Example for SNS-*m*-*n*

To illustrate the idea of SNS-m-n, we resort to the example network from Figure 2.18 and solve several instances of SNS-m-2 for different sizes of the sub-network, m. A sub-network must contain at least $n+1$ disjoint paths to be operational. The associated minimal value for m can be found by solving a min-cost-flow problem. The min-cost-flow problem moves $n+1$ units of supply from the source to the sink; the edge capacities are all one; and the cost of using an edge is one. The objective of the min-cost-flow problem gives the minimum number of edges required to guarantee a minimum cut of $n+1$ or greater in the SNS-m-n sub-network. The minimal size of the sub-network for the example is $m = 12$. The SPNI problem with two attacks in the original network has an objective function value of 13, which is the lower bound for any SNS-m-2. A sub-network with 21 arcs is sufficiently large to achieve the lower bound. Figure 2.21 shows the solutions for different SNS-m-2 problems. Figure 2.22 shows the objective function value as function of the size of the sub-network.

In addition to the small illustrative example, we solve SNS-m-2 for the German network, with Hamburg as source and Offenbach as sink. The shortest path between s and t consists of 55 arcs with a total length of 442,916 meters. The solution to SPNI with two attacks has length of 462,995 meters. A feasible solution is a sub-network consisting exclusively of three disjoint paths from s to t. The minimum number of arcs needed for such a network is 158. The length of the longest of the disjoint paths is used to shrink the network to size 537 nodes and 1,458 arcs. We solve SNS-200-2 in 16:07 hours. Figure 2.23 shows how the lower and upper bounds approach each other in the row-and-column generation approach as described in Section 2.4. Figure 2.24 shows the optimal sub-network for SNS-200-2.

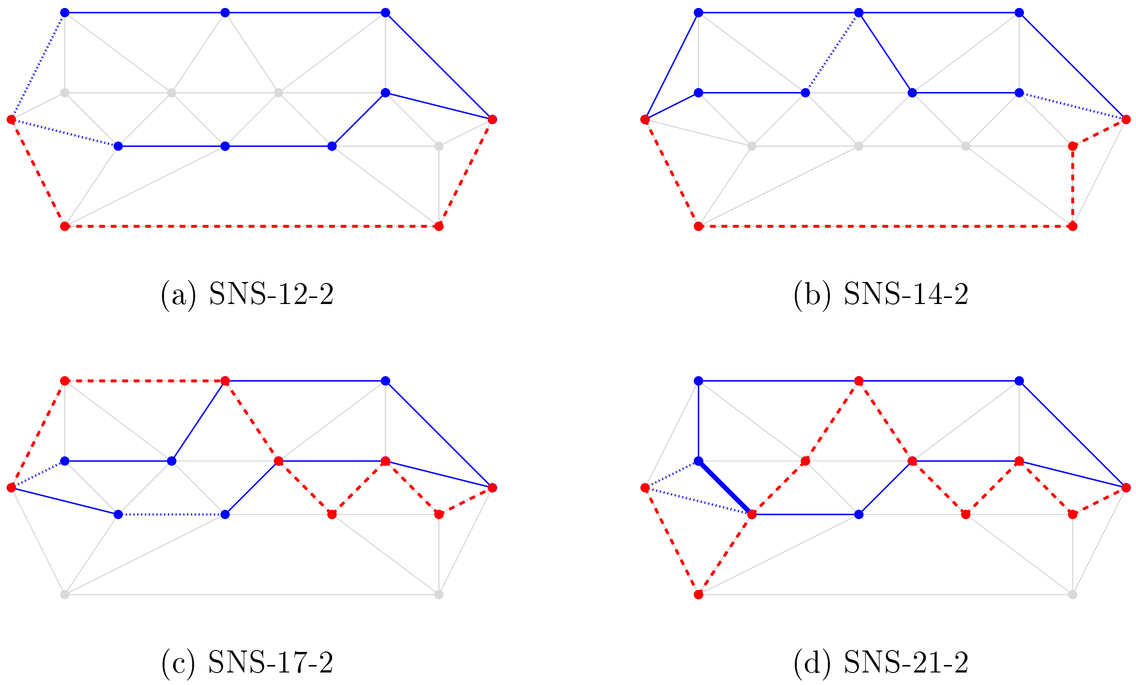


Figure 2.21: **SNS-m-2 example** - The plots show the sub-network, the optimal attack and the resulting shortest path for different instances of SNS-m-2. All the colored arcs are in the sub-network. The dotted blue arcs are attacked by the attacker and the dashed red arcs build the remaining shortest path after the optimal attack. (a) Solution to SNS-12-2, the minimal possible sub-network consists of three disjoint paths, where the two shortest are interdicted by the optimal attack. The remaining path has a length of 20. (b) Solution to SNS-14-2 with a length of the defender's response of 17. (c) Solution to SNS-17-2 with a length of the defender's response of 14. (d) Solution to SNS-21-2 where the defender is able to yield the lowest possible response of 13. SPNI on the original network and the sub-network have the same objective function value. Further increase of the sub-network size beyond 21 does not effect the defender's response to the worst-case attack on two arcs. The bold face arc in the solution indicates that the arcs for both directions are in the sub-network.

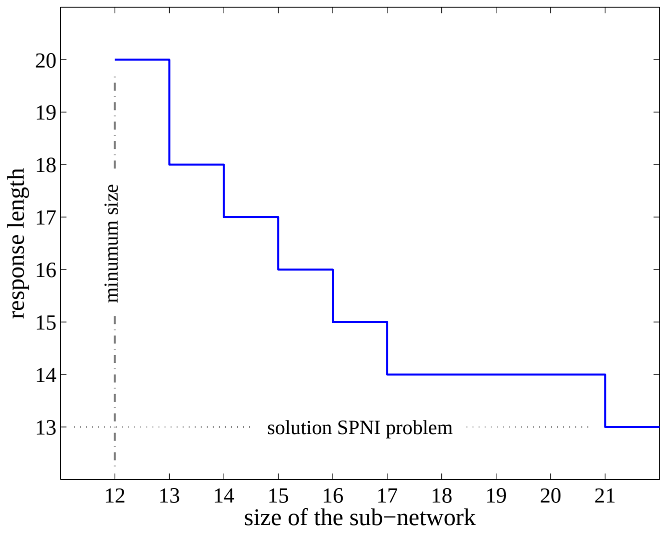


Figure 2.22: **SNS-m-2, objective function values** - The minimal size of the sub-network is 12. A sub-network with less than 12 arcs is non-operational. A sub-network with 21 arcs achieves the lower bound. Increasing m beyond 21 does not effect the length of the defender's response.

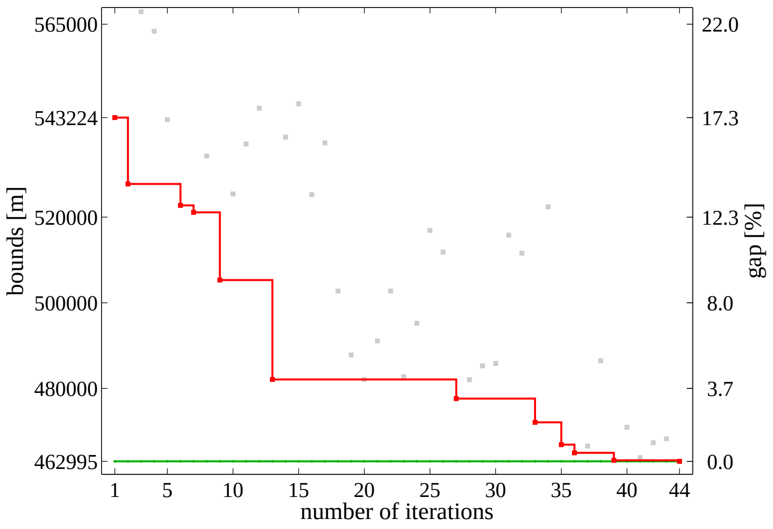


Figure 2.23: **Bounds for SNS-200-2 with the Germany network** - The plot shows the approaching lower and upper bounds during the 44 iterations of the row-and-column generation procedure. After finding a sub-network, we solve SPNI with two attacks on the sub-network. The gray dots are the resulting objective functions values. The lower bound remains constant because of an additional constraints that restricts the objective function value to be greater or equal to the solution for SPNI on the original network. Because of the constant lower bound, we can label the upper bound with the resulting optimality gap. The graph shows that a sub-network with 200 arcs is sufficiently large to yield the same result as the full network.

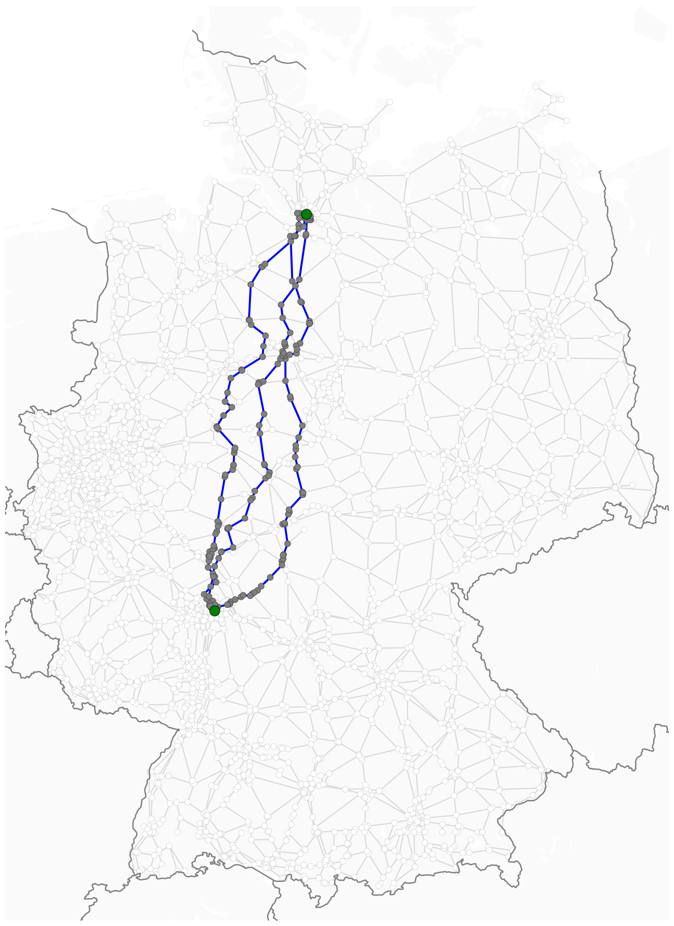


Figure 2.24: **Solution for SNS-200-2 with the Germany network** - The plot shows the optimal sub-network, consisting of 195 arcs. The defender's budget is not exhausted, telling us that the optimal sub-network with exactly 200 arcs does not perform better than the one shown. The SPNI problem for the shown sub-network and the SPNI for the original Germany network have the same objective, meaning that an increase in size of the sub-network does not affect the length of the defender's response.

The solution for the SNS-m-n example shown in Figure 2.24 contains disjoint paths that intersect each other, similar to the solutions for PPS-m-n. However, the performance is quite different. In PPS-m-n the defender preselects complete paths that are destroyed when attacked. The attacker's interdictions in SNS-m-n only destroy the single paths between intersections, because the defender is allowed to use the post-attack subgraph in any way he chooses. A direct comparison between the two problems is misleading because of the different interpretations of the defender's selection before the attack. A defender who selects about 200 arcs in SNS-m-n has more flexibility than a defender with about 200 arcs in their portfolio in PPS-m-n.

2.5.7 An Example for C-SNS-m-n

To illustrate the idea of C-SNS-m-n, we resort to the network from Figure 2.10 with its 2,971 nodes and 8,824 arcs and assume Hamburg as source and Munich as sink. The shortest path between the two nodes contains 76 arcs with a total length of 677,898 meters. With an attack budget of 150,000, we solve C-SNS-m-n for different sub-network sizes, m. The optimal sub-network from C-SNS-85-150000 does not provide more than one disjoint path and the defender has to suffer the full attack budget. The optimal sub-network for C-SNS-150-150000 provides two disjoint path from s to t. The attacker distributes the attack budget across the disjoint paths, such that they have equal length. Splitting the attack budget results in a smaller objective function value than the one for C-SNS-85-150000. Increasing m further can improve the sub-network's performance. The data for the different C-SNS-m-n problems are shown in Table 2.6.

m	# paths attacked	objective	runtime [min]
85	1	827,898	>60:00
150	2	761,475	02:45
250	3	753,156	00:02

Table 2.6: **Results for the C-SNS-m-n example** - We use the Germany network with its 2,971 nodes and 8,824 arcs; Hamburg is the assumed source and Munich serves a sink. We solve different instances of C-SNS-m-150000. Larger m values give the defender more options after the attack, and return lower objective function values. In the attacked optimal sub-network from C-SNS-250-15000, for instance, one can find three disjoint paths, each of which is attacked and of shortest length after the attack. The column labeled "# paths attacked" lists the number of disjoint paths that are attacked for each sub-network. Solving C-SNS-85-150000 was stopped after one hour. The objective function value for the best integer solution at this point was equal to the known best possible objective function value. The optimal sub-network consists of the 76 arcs from the shortest path in the original network and an arbitrary set of additional 9 arcs. Besides the large number of non-optimal solutions, there are many optimal solutions that one has to explore in the B&B tree. In addition, the solution for the continuous relaxation is poor, all resulting in the long solution time.

2.6 Conclusion

A Path-Portfolio-Selection problem is a three-stage sequential game, where a defender selects a number of paths from source to sink in a network, such that the shortest of those paths is of minimum length after an optimal attack. We define and distinguish various Path-Portfolio-Selection problems with different meanings of the attack budget and the attack outcome, that is, whether an attacked arc is destroyed or lengthened. This chapter creates models and practicable solution methods for PPS-m-n, where the attacker destroys n arcs in the defender's set of m s-t-paths. The models and solution methods developed generalize, with small changes, to variants of PPS-m-n: Pr-PPS-m-n, D-PPS-m-n, and C-PPS-m-n. The models and solution methods generalize further, to the Sub-Network-Selection problems SNS-m-n, Pr-SNS-m-n, D-SNS-m-n, and C-SNS-m-n. Finally, even though we concentrate on shortest path formulations, our models and solution methods could be extended to arbitrary minimum cost flow problems, significantly increasing the applicability of the results.

For PPS-m-n, we prove that only the longest path in the optimal portfolio is non-interdicted after the worst-case attack. Based on this result, we formulate the otherwise tri-level PPS-m-n as a MIP, which is theoretically solvable by standard solvers, but becomes intractable for large networks. Applying a row-and-column generation approach, adding supervalid inequalities, and solving a sequence of problems that successively remove superfluous parts of the network, we solve PPS-m-n for large networks in a reasonable time. We apply the proposed methods for a Germany road network with 2,971 nodes and 8,824 arcs and solve PPS-5-2 in about two hours, after identifying as unnecessary and removing more than 85% of nodes and arcs. The probabilistic version Pr-PPS-m-n can be solved in an similar way, requiring more computational effort. However, a near optimal solution with performance guarantee can be found quickly.

The defender's response path in a general D-PPS-m-n could be interdicted, which makes the problem difficult to solve. We provide a model that can find the optimal solution for such problems and apply the model on a small network as an illustration. For large networks, one can potentially find a similar way to remove unnecessary parts from the network and thereby reduce the run time. This could be a starting point for future work. We find the optimal portfolio for the continuous relaxation C-PPS-m-n by solving a single

MIP in a short amount of time, even for large networks. The solution, which is a set of disjoint paths with about equal and minimum length, makes the model potentially useful for other network problems.

We also propose solution models for the closely related SNS-m-n and all its variations. The defender in SNS-m-n has more flexibility, resulting in a better objective function value than the one for an equivalent PPS-m-n. Solution methods for SNS-m-n as well computational and theoretical results have strong similarities to PPS-m-n. However, the SNS-m-n models can be extended to other network optimization problems, such as network flow problems, which makes them particularly interesting. We define and solve example problems for SNS-m-n and its variants, but there is potential to improve the models and reduce solution times in a similar fashion as our work demonstrates for PPS-m-n, which is left for future work.

CHAPTER 3: Augmenting a Stochastic Logistics Network with Warehouses to Optimize Reliability

3.1 Abstract

Humanitarian assistance cargo is shipped by the Department of Defense using uncertain space available on regularly scheduled transportation routes for ships and aircraft. The uncertainty of available space suggests a directed stochastic network model, where arcs, representing the transportation routes, can fail randomly. Currently, cargo cannot be stored in transshipment nodes, i.e., it cannot be stored at an airport or seaport for further transportation. If the cargo's origin and destination are disconnected in the stochastic network, shipment must be delayed. We define expected shipping time as the average time cargo must wait from its generation at the source node until it reaches its destination. Actual transit time is negligible compared to the waiting time and is ignored.

The research objective is to minimize the expected shipping time by adding storage capacity (warehouses) to some of the transshipment nodes. We provide a method to find the optimal location for one and two warehouses, involving a specialized sampling method to estimate s-t-reliability. Our method uses novel necessary inequalities to eliminate many possible solutions from consideration, saving on computational effort. Furthermore, we give a bound on the shipping time based on an unlimited number of warehouses. For a real-world network between the U.S. and Europe that bound shows that adding more than two warehouses provides little additional benefit.

3.2 Introduction

3.2.1 Motivation

Humanitarian assistance is of growing importance to the United States, and is shipped by the Department of Defense using space available on existing military transportation routes [32]. A transport on a transportation route between two military bases is carried out by ship or aircraft according to a recurring regular schedule. The collection of existing transportation routes forms a logistics network.

A network operator has access to the data of all regularly scheduled military transports and can identify space available. The data are valid for a certain amount of time, called a "round", and changes afterward. Only space available can be used to ship humanitarian assistance cargo from its source to its destination, and cargo cannot be stored at any transshipment locations. To ship cargo, the operator needs to identify a sequence of routes from source to destination in the current round, each route providing space available. Until such a sequence of routes exists, shipment of existing humanitarian assistance cargo is delayed.

Dozier and Dimitrov [32] analyze several years of space-available data in the European area of operations, and find that there is insufficient capacity to carry humanitarian cargo in a reasonable amount of time. The European area contains 17 sink locations for humanitarian assistance cargo, which can originate at any military base within the United States. The transportation routes are distinguished as U.S. Air Force and U.S. Navy routes. The time horizon of a round in Dozier's data set is two weeks.

From the given data set we can compute a probability that space is available within a round for each transportation route. A transportation route not offering space available in a particular round is not available for shipping, and can be treated as non-existing in that round. Therefore, the transportation network has a stochastic nature, and one can compute the expected number of rounds required until the stochastic network contains a path from source to sink. That number represents the time until the cargo is shipped. Adding storage capacity, warehouses, to some of the transshipment nodes has the potential to decrease the time until cargo is shipped, raising questions about the optimal locations to add storage capacity. The *Warehouse Location Network Reliability* problem (WLNR) seeks to identify the optimal warehouse locations. We develop a case study using all U.S. Air Force flights and associated data about space available extracted from Dozier's data set. The resulting stochastic logistics network consists of 95 nodes and 1,096 arcs, and is shown in Figure 3.1.

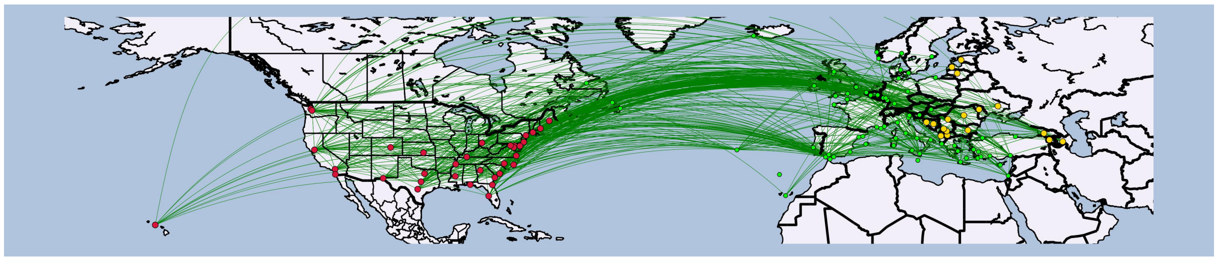


Figure 3.1: **U.S. Air Force routes from Dozier's data set** - The network consists of 95 nodes and 1,096 arcs. Each arc represents a transportation route operated by the U.S. Air Force according to a regular recurring schedule. Each route has an associated probability that space is available within a time horizon of two weeks.

3.2.2 Background and Related Work

Research on network reliability and warehouse allocation problems is relevant to WLNR. Warehouses provide storage capacity and their location can be critical for business. A solution to a warehouse allocation problem recommends locations for warehouses that help meet demand with high probability, minimize operating costs, maximize customer satisfaction, etc. As one of the first appearances in the literature, Baumol and Wolfe [33] investigate the number and geographical locations of warehouses for a large corporation. They explore the problem formulation and give a solution procedure. In general, allocation problems are concerned with placing facilities, and warehouses are special facilities. Literature is filled with a variety of facility location problems that pursue different objectives. For a more detailed review, we refer to Drezner and Hamacher [34], and Owen and Daskin [35]. In general, facility location problems are assignment problems and vary from linear assignment problems, also known as bipartite weighted matching problems [19], to more complicated and hard-to-solve quadratic assignment problems [36]. WLNR is an assignment problem; it assigns a warehouse to some of the transshipment nodes.

The main difference between facility location problems and the WLNR problem is the stochastic nature of the network. Facility location problems typically assume a fixed deterministic network. Uncertainty is often considered in supply and demand, but not in route failures. Mirchandani and Odoni formulate one of the rare examples that relaxes the deterministic network assumption [37, 38]. They extend the m-median problem to a stochastic network, and treats the arc lengths as random variables with known probability distributions. The m-median problem seeks to design a network with minimum average travel cost by identifying m optimal facility locations. An illustrative example is to place m fire units such that the average travel time to a fire is minimized. Weaver and Church show a way to solve the two-facility location problem for Mirchandani's stochastic network, which consists of only ten nodes [39]. To the best of our knowledge, we are the first to study the WLNR problem.

Random arc and node failures can disrupt the functionality of a network. The reliability of a network is the ability to cope with random failures while serving the network's purpose. The most general network reliability analysis problem is the k-terminal problem. The measure for the k-terminal problem is the probability that there exist operating paths from the source to each node in a set of nodes, K, where $k = |K|$. Ball [11] discusses five conditions, such that if any one of those is true, the reliability analysis problem is NP-hard. A special case of the k-terminal problem is the s-t-reliability problem, also known as two-node-reliability. The s-t-reliability for a stochastic network is the probability that source, s, and sink, t, are connected. Valiant [12] proves that calculating the s-t-reliability belongs to the class of #P-complete problems. We note that calculating the nominal shipping time amounts to calculating reciprocal of the s-t-reliability between source and sink. That fact links to the complexity of the WLNR problem. (See Lemma 5.)

There are practicable procedures available to calculate s-t-reliability for small networks. However, calculating s-t-reliability for larger networks can be difficult and time consuming, and therefore people try to bound the probability that s and t are connected. Ball and Provan [13] restrict the problem to a common arc failure probability p, and give bounds in terms of polynomials in p. The literature is filled with different methods to tackle the problem for the general case, where arcs fail with arc-dependent probabilities. The core of all the bounding methods is one of the following three major techniques:

- consider only paths up to a certain length (provides lower bound) [14],
- consider only a subset of nodes (provides upper bound) [14], or
- find disjoint success and failure events (provides lower and upper bounds) [15].

Crude Monte Carlo sampling, however, can estimate s-t-reliability quickly, though it does not provide bounds that hold with certainty. A stochastic network has $2^{|A|}$ different network states. That number makes it intractable to check all states for connectedness between source and sink. Therefore, the s-t-reliability is estimated using r randomly chosen samples of the network's state space. The sampling error for independent replications decreases as $1/\sqrt{r}$, and thus more samples mean smaller error. The cost for a crude Monte Carlo sampling experiment depends on r and the cost to draw and evaluate a sample of the stochastic network.

The literature discusses several specialized sampling methods to obtain better Monte Carlo estimates while keeping the number of replications constant. One of these methods is importance sampling [40, 41]. Fishman proposes and compares different variance reduction methods for estimating s-t-reliability [42, 43, 44]. Van Slyke and Frank [45] propose stratified sampling methods to yield computational savings. However, the special structure of the WLNR problem renders our sampling procedure preferable.

3.2.3 Problem Statement and Organization

Throughout the paper, we make the following assumptions and simplifications to the general problem. From all the source and sink locations for humanitarian assistance cargo, we always pick one fixed source-sink pair for analysis and numerical experiments. The solution for multiple source and multiple sink problems can be derived from considering all possible source sink pairs in the problem. At the end of the paper, we solve WLNR for Dozier's data set considering multiple sources and sinks.

Furthermore, we only consider humanitarian assistance cargo and space available with a constant size of one unit. Logistics often defines its own cargo size depending on the problem. The smallest unit of cargo in Dozier's data set is one pallet. For simplicity, we assume each humanitarian assistance cargo requires space available of one pallet.

The warehouse location network reliability (WLNR) problem considers a directed, stochastic network, $G = (N, A)$, with set of nodes N and set of arcs A, where $a = (i, j) \in A \subset N \times N$. *Source* and *sink* are distinguished nodes and represent origin and destination of the cargo, respectively. Each arc $a \in A$ is independently in one of two states. It is present, with probability p_a, or not present, with probability $1-p_a$. All the arc states at a given time define a *network state*. The network state changes every *round*, where a round represents a fixed length of time, for example, two weeks in Dozier's data set.

Cargo needs to be shipped from s to t across present arcs and cannot be stored at any of the transshipment nodes. If the network does not provide a path from source to sink in the current round, shipment is delayed. The *shipping time* is the expected number of rounds required to ship the cargo. The time the cargo is *en route* is negligible, compared to the waiting time until a path from s to t is available. So actual en-route time is ignored when computing shipping time. In other words, no matter the length any path through the network is equally good.

Adding storage capacity, warehouses, to some of the transshipment nodes can decrease the expected shipping time below the *nominal shipping time* with no warehouses in place. A warehouse can store cargo indefinitely. Given a budget to build n warehouses, the optimal solution to the WLNR problem places the warehouses such that the shipping time is minimized. Figure 3.2 depicts the idea for a small example network.

Section 3.3 explores solution methods involving a Markov Decision Process with Design. The complexity of the model quickly reaches the limits of tractability, but the model provides insight in the problem that leads to a Markov chain in Section 3.4. The Markov chain approach returns formulas for shipping times that are easy to evaluate for each warehouse location, suggesting a brute force approach. However, computing or estimating the needed s-t-reliabilities is computationally expensive. In Section 3.4.2 we show a method to solve WLNR for one warehouse, which is computationally close to brute force. However, computational complexity to solve WLNR with two warehouses can be lowered a great deal as shown in Section 3.4.3. Section 3.4.4 provides a bounding method, which is useful to estimate the gain of adding another warehouse to the network. Section 3.5 gives insight into the specific methods we used to simulate more efficiently, and a solution for the case study network is provided in Section 3.6.

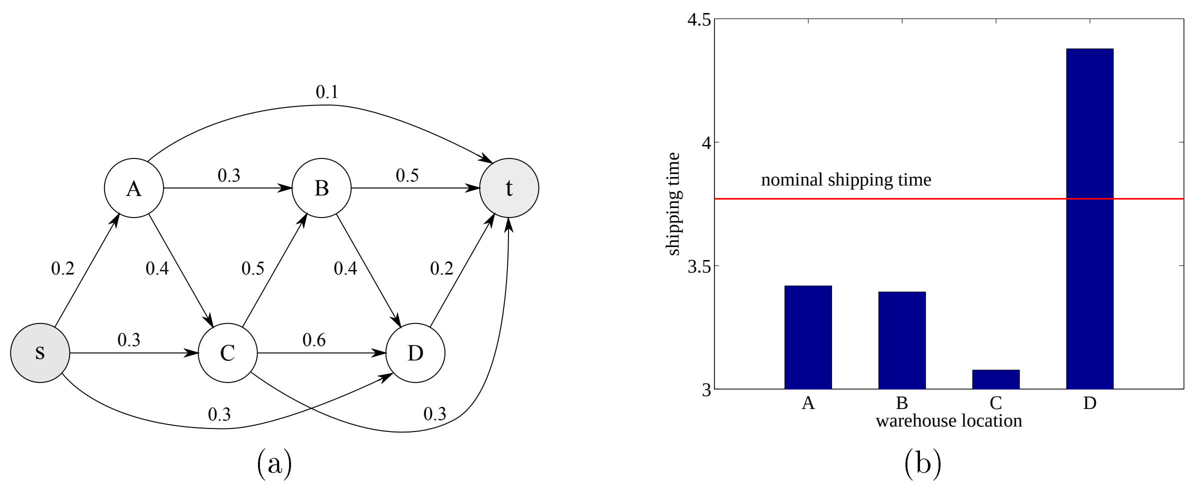


Figure 3.2: **Stochastic network and solution for placing one warehouse** - (a) The network, where s is the source and t is the sink node. The label on each arc gives the probability that the particular arc is present. The objective is to minimize the expected shipping time between s and t by adding a single warehouse to the network. (b) Expected shipping time for each possible warehouse location including the nominal shipping time as reference. Node C is the optimal place to put the warehouse. Placing the warehouse at node D would increase the shipping time beyond the nominal shipping time.

3.3 Markov Decision Process Approach

The solution to the WLNR problem consists of two parts. First, it optimally allocates the warehouses, and second, it defines the associated optimal shipping policy. The shipping policy is a rule describing where to ship the cargo for each state of the system. The *state of the system* at any round is defined by the state of the network and the cargo's location. Because of the assumption that traversing a present arc in the stochastic network takes no time, the cargo can only be located at nodes with storage capacity (i.e., source, warehouses and sink). Solving a *Markov Decision Process* (MDP) for given warehouse locations yields the shipping policy with minimum shipping time. Formulation and solution approaches for a MDP are shown in [46].

The problem of selecting both warehouse locations and shipping policy can be modeled as a *Markov Decision Process with Design* (MDPD); a specific example is presented by Dimitrov and Morton [47]. Here the design part comes from the warehouse allocation, and one can envision the MDPD as a two-stage solving process, but the two stages are solved simultaneously. In the first stage, the warehouses are placed, and in the second stage, a MDP is solved to find the optimal shipping policy. State space and set of possible actions for the MDP is defined by the decision in the first stage. The following formulation for MDPD, though large, can be solved in theory using a standard mixed integer programming solver.

Sets

$i, j \in N$	nodes in network G; s an t are distinguished as source and sink
$(i, j) \in A$	arcs in G
$w \in W$	all possible warehouse locations, including s and t
$\phi \in \Phi$	ϕ is a particular realization of the stochastic network with arcs turned on or off randomly; Φ, the union of all possible ϕ, is the state space of the network
$\psi \in \Psi$	ψ is a state of the MDPD, defined by a pair $(\phi(\psi), w(\psi))$, with $\phi(\psi) \in \Phi$, a network state, and $w(\psi) \in W$, a warehouse where the item that is to be shipped is located; Ψ is the state space of the MDPD
$a_\psi^w \in \mathcal{A}_\psi$	the action a_ψ^w ships the item from warehouse $w(\psi)$ to warehouse w; all actions one can execute in state ψ are contained in $\mathcal{A}_\psi$

Data

n	number of warehouses to add to the network
M	a large positive value
ξ_ψ	probability that the system starts in state ψ; ξ_ψ carries the information about the source; that is $\xi_\psi > 0$ only if $w(\psi) = s$
$p(\phi)$	probability that the network is in state $\phi \in \Phi$
$P(\psi\|\psi', a)$	probability that the system in state ψ' makes a transition into state ψ by executing action $a \in \mathcal{A}_{\psi'}$; $P(\psi\|\psi', a)$ carries information about the sink; that is, $P(\psi\|\psi', a) = 0$ if $w(\psi') = t$; the cargo disappears once it arrives at the sink

Variables

v_ψ	value of the system in state ψ, which is the expected number of rounds needed to ship the item to the sink, if the system is in state ψ
v_w	value of the item at location w, which is the expected number of rounds needed to ship the item to the sink
$x_{\psi a}$	expected number of times action a is performed in state ψ
z_w	1 if a warehouse is located at $w \in W$ is selected, and 0 otherwise

MDPD, primal formulation

$$\min_{\boldsymbol{x},\boldsymbol{z}} \sum_{\psi\in\Psi} \sum_{a\in\mathcal{A}_\psi} x_{\psi a} \tag{3.1a}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}_\psi} x_{\psi a} = \xi_\psi + \sum_{\psi'\in\Psi} \sum_{a\in\mathcal{A}_{\psi'}} p(\psi|\psi', a)\, x_{\psi' a} \qquad \forall \psi \in \Psi \tag{3.1b}$$

$$\sum_{w\in W} z_w \le n + 2 \tag{3.1c}$$

$$x_{\psi a_\psi^w} \le M z_w \qquad \forall \psi \in \Psi,\ w \in W \tag{3.1d}$$

$$z_s, z_t = 1 \tag{3.1e}$$

$$z_w \in \{0,1\} \qquad \forall w \in W \tag{3.1f}$$

$$x_{\psi a} \ge 0 \qquad \forall \psi \in \Psi,\ a \in \mathcal{A}_\psi \tag{3.1g}$$

The variable $x_{\psi a}$ represents the expected number of times that action a is performed if the system is in state ψ. The sum of all $x_{\psi a}$ is the expected number of times that any action is performed. An action is performed as long as the cargo has not arrived at the sink node. Thus, the objective function value (3.1a) is the expected number of rounds needed to ship the cargo from the source to the sink. The vector ξ_ψ defines the source as the cargo's initial location. Equation (3.1b) is the balance equation: the expected number of times leaving a state equals the expected number of times entering that state, including inflow from the initial probability mass ξ_ψ. We can select n warehouses, where selecting a warehouse allows the associated variables $x_{\psi a}$ to obtain a positive value. An unselected warehouse cannot receive shipments and corresponding $x_{\psi a}$ are forced to be zero by (3.1d). Source and sink are treated as warehouse locations. Initially, the cargo is located at the source and finally at the sink. Although, the variables z_s and z_t will be forced to be one, we add the redundant constraints (3.1e), and account for that by allowing a budget of $n+2$. The budget constraint is expressed in (3.1c). The optimal shipping policy can be computed from $\boldsymbol{x}$. For any state of the system ψ, there is only one $x_{\psi a}$ that is non-zero, pointing to the action that be performed.

We use the MDPD from model (3.1) to find the computational complexity for solving WLNR.

Lemma 5. *The WLNR problem is #P-hard.*

Proof. We show that finding the optimal solution for model (3.1) is #P-hard. Assume a warehouse budget of zero. Solving model (3.1) returns the optimal solution for WLNR with zero warehouses. The objective function value is the expected number of rounds needed to ship the item from source to sink. Because of geometric behavior, the reciprocal value is the probability that there is a path from source to sink, the *s*-*t*-reliability, which is proven to be #P-complete [12]. □

The dual variables for the continuous relaxation of the MDPD formulation have meaningful interpretations. For each balance equation (3.1b), we define a dual variable v_ψ, which represents the expected number of rounds needed to ship the cargo to the sink, given the system in state ψ. Though the binary variables $\boldsymbol{z}$ in (3.1f) prevents us from taking the dual, the MDPD can be formulated using the dual variables as a min-max problem as follows:

MDPD, formulation with dual variables

$$\min_{\boldsymbol{z}} \max_{\boldsymbol{v}} \sum_{\psi \in \Psi} \xi_\psi v_\psi \tag{3.2a}$$

$$\text{s.t.} \quad v_w = E_\phi[v_{(\phi,w)}] = \sum_{\phi \in \Phi} p(\phi)\, v_{(\phi,w)} \qquad \forall w \in W \tag{3.2b}$$

$$v_\psi - M(1 - z_{w(\psi)}) \leq 1 + v_{w'} + M(1 - z_{w'}) \quad \forall \psi \in \Psi,\ a_\psi^{w'} \in \mathcal{A}_\psi \tag{3.2c}$$

$$\sum_{w \in W} z_w \leq n + 2 \tag{3.2d}$$

$$v_{(\phi,w)} \leq M z_w \qquad \forall w \in W,\ (\phi, w) \in \Psi \tag{3.2e}$$

$$v_\psi = 0 \qquad \forall \psi : w(\psi) = t \tag{3.2f}$$

$$z_s, z_t = 1 \tag{3.2g}$$

$$z_w \in \{0, 1\} \qquad \forall w \in W \tag{3.2h}$$

As in the primal model, the objective function value is the expected number of rounds required to ship the cargo from source to sink. Constraint (3.2c) sets the value of the system that is in state ψ. Cargo at location $w(\psi)$ experiences an additional shipping time of $1 + v_{w'}$ if shipped to location w' in the current round. The minimum additional waiting time is achieved by shipping to the appropriate w'. If at least one of the locations $w(\phi)$ or w' has no storage capacity, then constraint (3.2c) is vacuous by the use of the big-M value. The cargo's value at a warehouse w can be computed from all values of the system, where the cargo is located at warehouse w, as shown in constraint (3.2b). To make it undesirable to ship cargo to a location that has no storage capacity, we force the associated $v_{(\phi,w)}$ to zero by (3.2e). The cargo's value at the sink is zero by (3.2f), because it takes zero rounds to ship from the sink to itself. We select source and sink as warehouses by (3.2g) and account for that in the budget constraint (3.2d).

Model (3.1) is a complete mathematical formulation for WLNR, but the size of the state space, $|\Psi| = |A| \cdot 2^{|A|}$, makes the formulation intractable for practical implementation. The dual formulation in model (3.2) is not solvable because some of the variables minimize and others maximize. However, for fixed warehouse locations, the inner maximization problem provides the intuitively optimal policy, which ships the cargo to the reachable warehouse w with smallest v_w. This result leads to a new model. We take the nodes with storage capacity and add directed arcs, (i, j), such that $v_j \leq v_i$. We call such a graph an abstract network representation, or abstract model, for short. For that model, the MDP behaves like a Markov chain.

3.4 Markov Chain Approach

An abstract model for a network with two warehouses is shown in Figure 3.3. All possible shipments are defined by the arcs in the abstract model. The model is constructed such that $v_s > v_v > v_w > v_t$. Thus, shipping to the node that is as far to the right as possible reflects the optimal shipping policy. We assign probabilities to the arcs that capture the desired shipping policy. For example, the probability of shipping from s to v is the probability that there is a path from s to v available, but there is no path from s to w, nor a path from s to t at the same time. We use the following definitions and notation to fully describe the abstract model and associated Markov chain.

Notation

$P_{xy}^{\overline{z}_1 \dots \overline{z}_m}$	probability that there is at least one path from warehouse x to warehouse y available and there is no path available from x to any of the warehouses $z_1 \dots z_m$
(s, v, w, t)	a shipping scheme with source node s, sink node t and warehouses $\{v, w\}$, imposing the following rules: (1) only ship from a location to any other location listed to the right in the scheme, (2) ship as far as possible to the right
$E_T(\text{scheme})$	expected shipping time following the shipping scheme as specified

An abstract model, e.g., the model in Figure 3.3, is a Markov chain not including self loops. A *self loop* is an arc starting and ending at the same node. The probabilities of traversing a self loop can be computed from the fact that, for each node, the sum of the probabilities of outgoing arcs equals one. Computing the expected shipping time in the Markov chain from a node n to t gives exactly the value v_n. To complete the model, we need to find the probabilities assigned to each arc. Each of the probabilities in Figure 3.3 is a s-t-reliability, which can be calculated, bounded, and estimated.

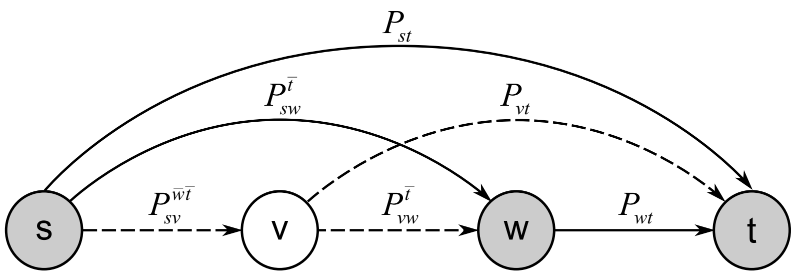


Figure 3.3: **Abstract network representation with two warehouses** - The drawing only contains nodes that have storage capacity, i.e., the source (s), the first warehouse (v), the second warehouse (w), and the sink (t). The nodes are ordered from left to right according to the expected shipping time from the node to the sink, e.g., expected shipping time from w to t is smaller than the expected shipping time from v to t. The arcs represent possible shipments, i.e., only shipments between specific warehouses are possible. Other shipments, e.g., from w to v are not considered in the shipping rule, which ships to a node as far to the right as possible. The model is expressed by the shipping scheme (s, v, w, t). An abstract network representation with one warehouse considers only the grayed nodes and solid arcs, with shipping scheme (s, w, t).

Calculating s-t-reliability is known to be a #P-complete problem [12], and there are methods available to calculate and bound s-t-reliabilities. For now, let us assume we have a black box that returns the s-t-reliability for a stochastic network with specified nodes s and t. The black box can find the probabilities P_{st}, P_{vt}, P_{wt} from Figure 3.3 directly but cannot calculate the remaining probabilities, e.g., $P_{sw}^{\bar{t}}$. However, this probability can be computed as

$$P_{sw}^{\bar{t}} = P_{s(w,t)} - P_{st},$$

where $P_{s(w,t)}$ is the probability that there is a path available form s to w or a path available from s to t in the stochastic network. If one adds the permanent arc (w, t) to the network, then $P_{s(w,t)}$ equals P_{st}, which can be found directly by the assumed black box. The idea is depicted in Figure 3.4.

3.4.1 Calculating and Estimating Reliabilities

The previous section assumes a black box that computes s-t-reliabilities. This section discusses different methods that could fill the black box and produce the desired output. The literature provides several methods to calculate or bound s-t-reliabilities, and each method is typically applied to a small network (about 20 nodes and $50, \ldots, 100$ arcs) for evaluation purpose. We used the Equivalent Link Algorithm (ELA) [15] to build our

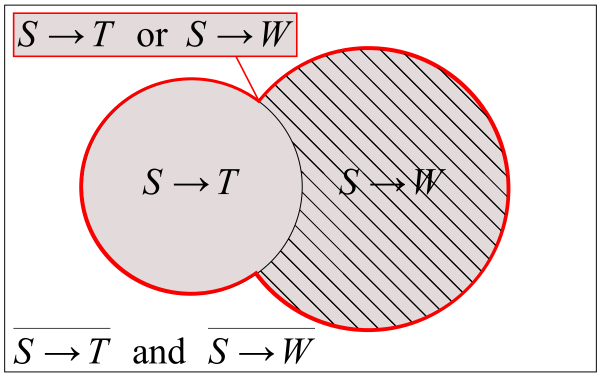

$S \rightarrow T$ or $S \rightarrow W$
$S \rightarrow T$
$S \rightarrow W$
$\overline{S \rightarrow T}$ and $\overline{S \rightarrow W}$

Figure 3.4: **Venn diagram depicting the idea for calculating missing probabilities** - The interior of the red bounded shape represents all the network states, including a path from s to t or a path from s to w. From that set of states, we subtract all network states that have a path from s to t included. The number of states in the remaining shaded set can be used to calculate the probability $P_{sw}^{\bar{t}}$.

black box and solved the example shown in Figure 3.2. The runtime of the algorithm increases exponentially with an increasing number of paths in the network. ELA, as well as other proposed methods, are not suitable to calculate or bound s-t-reliabilities for Dozier's transportation network because of network size.

Besides calculating and bounding s-t-reliabilities, we can estimate reliabilities. To find the s-t-reliability for a stochastic network, one can perform the following procedure. First, identify the success states, $\phi_+ \in \Phi$, that are the network states including a path from s to t. Second, calculate the state probabilities $p(\phi_+)$, and sum the state probabilities for all success states. Here we face the same problem that prevents us from using the MDPD approach. The size of the state space, $|\Phi| = 2^{|A|}$, makes this method intractable for reasonable sized networks.

Monte Carlo sampling considers $r \ll |\Phi|$ randomly chosen samples of the network's state space, including possible duplicates. A sample is successful if it belongs to the set of success states. The s-t-reliability can be estimated by the proportion of successful samples drawn, not providing bounds that hold with certainty. The cost for a Monte Carlo experiment depends on three factors: (1) the number of samples, (2) the cost of drawing one sample from the network's space state, and (3) the cost to identify whether a sample is a success.

The width of a confidence interval, CI, is a measure for accuracy of an estimate. A commonly used confidence level is 95%, which says that the CI covers the estimated parameter in 95% of the conducted experiments. Devore [48] provides the formula to calculate the $100(1-\alpha)\%$ CI for population proportions. The formula shows that the width of the CI decreases as $1/\sqrt{r}$. Thus the error can be controlled by the number of samples. Picking a confidence level (we pick $\alpha = 0.05$) and assuming a value for the unknown population proportion allows us to calculate the number of samples required to keep the width of the CI within a desired limit. People use $\frac{1}{2}$ as a conservative assumption for the unknown probability. We decide to keep half the width of the confidence interval within 5% of the point estimate's value, and divide the width of the CI by the assumed point estimate. Now, the number of samples required increases with decreasing assumed population proportion. The value $\frac{1}{2}$ is no longer a conservative assumption. We use a value of 0.0005, one of the smallest s-t-reliabilities between two nodes in the Air Force network. The influence by the number of samples on the CI's width is shown in Figure 3.5, which suggests to sample at least three million times.

Variance reduction methods can reduce the variance of a Monte Carlo experiment, which decreases the CI's width, keeping the number of samples constant. In other words, specialized sampling methods can provide the same width of the CI while sampling less. There are variance reduction methods available for the two-node-reliability problem and k-node-reliability problems. Such methods are specialized to estimate a single probability, i.e., the probability that two nodes are connected and the probability that there is a path from a start node to each of the k target nodes, respectively. To solve WLNR, we need to estimate more than one probability and each network sample can be used for multiple estimates. Therefore, sampling methods discussed in [42, 44, 43, 45] are not used here.

3.4.2 Adding a Single Warehouse

With reference to Figure 3.3, consider the gray nodes and incident solid arcs only. The shipping time for any warehouse location, w, is computed as:

$$E_T(s,w,t) = \frac{1 + \dfrac{P_{sw}^{\bar{t}}}{P_{wt}}}{P_{st} + P_{sw}^{\bar{t}}}. \tag{3.3}$$

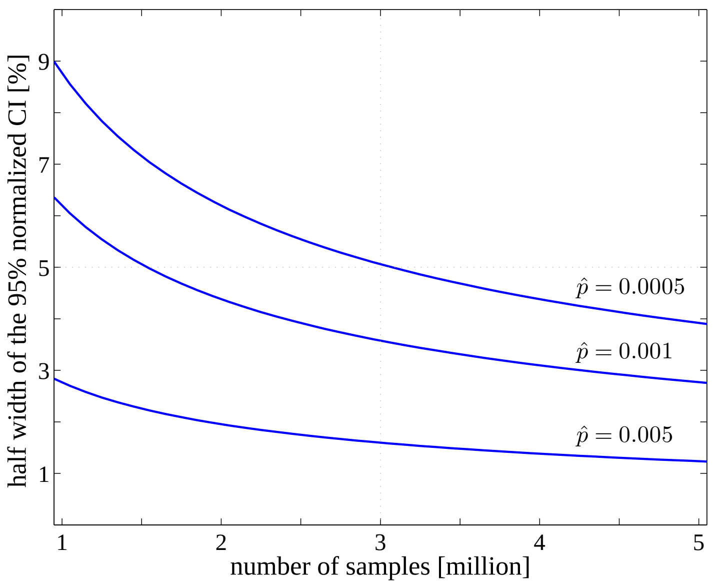


Figure 3.5: **CI influenced by the number of samples** - The figure shows half of the width of the 95% CI for the population proportion as a function of the number of samples taken in a Monte Carlo sampling experiment. The width of the CI is normalized with the assumed population proportion, $\hat{p}$. The graph shows that at least three million samples are necessary to keep the width of the CI within $\pm 5\%$, for population proportions as small as 0.0005.

The solution to WLNR with one warehouse minimizes (3.3) with respect to the warehouse location. Changing the location for warehouse w clearly impacts the probabilities $P^{\bar{t}}_{sw}$ and P_{wt}, but in a way we cannot capture as a function of w. One way to solve the problem is to calculate the shipping time for every possible warehouse location and select the warehouse with resulting smallest shipping time.

Adding a single warehouse to the network and following the defined shipping rules does not always decrease the shipping time. Figure 3.2 shows an example where the warehouse at location D increases the shipping time beyond the nominal shipping time. We define the set of *good warehouse locations* $W_G \in N \backslash \{s, t\}$ as all nodes, such that $E_T(s, t) > E_T(s, w, t)$ $\forall w \in W_G$. In other words, a single warehouse installed at location $w \in W_G$ decreases the shipping time below the nominal shipping time. The property that distinguishes the good warehouse locations from the others is the subject of Lemma 6.

Lemma 6. *$E_T(s, w, t) < E_T(s, t)$ if and only if $P_{wt} > P_{st}$.*

Verbalized, a warehouse location is a good warehouse location if and only if the expected shipping time from the warehouse is smaller than the nominal shipping time.

Proof.

$$E_T(s,w,t) = \frac{1 + \dfrac{P_{sw}^{\bar{t}}}{P_{wt}}}{P_{st} + P_{sw}^{\bar{t}}} < \frac{1}{P_{st}} = E_T(s,t)$$

$$1 + \frac{P_{sw}^{\bar{t}}}{P_{wt}} < 1 + \frac{P_{sw}^{\bar{t}}}{P_{st}}$$

$$P_{wt} > P_{st}$$

□

Lemma 6 can be used to save on computational effort. A warehouse location w is rendered suboptimal if $P_{wt} > P_{st}$, no matter the probability $P_{sw}^{\bar{t}}$. There is no need for computing or estimating $P_{sw}^{\bar{t}}$ for a suboptimal w.

3.4.3 Adding Two Warehouses

After adding a single warehouse, one could wish to further improve the network performance and consider adding two warehouses to the network. While finding the best single warehouse location requires the calculation of $|W|$ shipping times, we now face the problem of calculating the shipping times for $(|W|^2 - |W|)$ combinations of two warehouses. The equation for the shipping time with two warehouses is shown in (3.4), which is associated with the Markov chain in Figure 3.3.

$$\begin{aligned} E_T(s,v,w,t) &= \frac{1 + P_{sv}^{\overline{wt}} E_T(v,w,t) + P_{sw}^{\bar{t}} E_T(w,t)}{P_{sv}^{\overline{wt}} + P_{sw}^{\bar{t}} + P_{st}} \\ &= \frac{1 + P_{sv}^{\overline{wt}} \dfrac{1 + \dfrac{P_{vw}^{\bar{t}}}{P_{wt}}}{P_{vw}^{\bar{t}} + P_{vt}} + \dfrac{P_{sw}^{\bar{t}}}{P_{wt}}}{P_{sv}^{\overline{wt}} + P_{sw}^{\bar{t}} + P_{st}} \end{aligned} \tag{3.4}$$

Solving the WLNR problem for one warehouse previously provides already estimates of P_{st} and $P^{\bar{t}}_{sw}$ for all w. To evaluate (3.4), the values of $P^{\overline{wt}}_{sv}$ and $P^{\bar{t}}_{vw}$ for each combination of v and w are needed in addition. One has to estimate another $2|W|^2 - 2|W|$ probabilities, which can be time consuming.

Theorem 2. *The optimal shipping scheme for a network with two warehouses has the following properties:*

- $E_T(s,w,t) < E_T(s,t)$ *and*
- $P_{vt} < P_{wt}$.

Verbalized, the warehouse w in Figure 3.3 must be a good warehouse location, and the probability of shipping from v directly to the sink is less than the respective probability from w.

While the work needed to evaluate (3.4) for each warehouse combination is negligible, estimating the required probabilities can take a remarkable amount of time. Theorem 2 identifies some combinations of warehouses from the solution space as suboptimal. We do not consider those suboptimal warehouse combinations and therefore do not have to estimate the associated probabilities, which is the power of the theorem. After solving the WLNR problem with one warehouse as proposed, we have access to all information needed to identify suboptimal warehouse combinations, limiting the number of solutions for the WLNR problem with two warehouses.

Proof for Theorem 2

The abstract representation for the network with two warehouses is shown in Figure 3.6. The model only includes nodes with storage capacity. Node one represents the source; we call node two the first warehouse, node three the second warehouse, and node four is the sink. We define the set of good warehouse locations $W_G \in N\backslash\{s,t\}$ as all nodes, such that $E_T(1,4) > E_T(1,w,4)\ \forall w \in W_G$. And, as a reminder, we restate the already proven Lemma 6.

$$E_T(1,3,4) < E_T(1,4) \quad \text{if and only if} \quad P_{34} > P_{14} \tag{3.5}$$

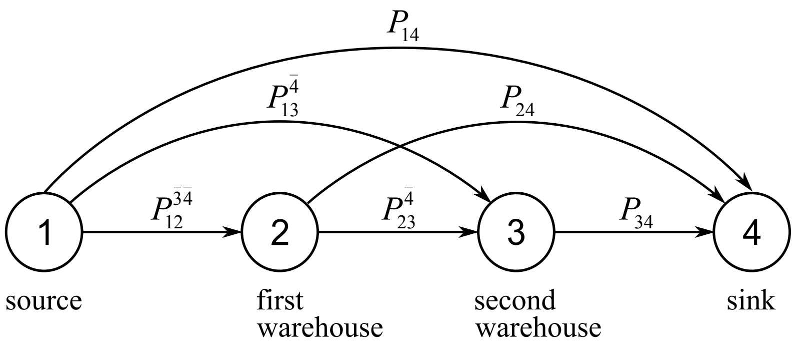


Figure 3.6: **Abstract network representation with two warehouses** - The drawing contains only nodes that have storage capacity, e.g., the source, the first warehouse, the second warehouse and the sink. The second warehouse belongs to the set of good warehouse locations if $P_{34} > P_{14}$.

The shipping time associated with the model in Figure 3.6 is calculated as shown in (3.6),

$$\begin{aligned} E_T(1,2,3,4) &= (1 - P_{12}^{\bar{3}\bar{4}} - P_{13}^{\bar{4}} - P_{14})(E_T(1,2,3,4) + 1) \\ &+\ P_{12}^{\bar{3}\bar{4}}(E_T(2,3,4) + 1) \\ &+\ P_{13}^{\bar{4}}(E_T(3,4) + 1) \\ &+\ P_{14} \\ (P_{12}^{\bar{3}\bar{4}} + P_{13}^{\bar{4}} + P_{14})E_T(1,2,3,4) &= P_{12}^{\bar{3}\bar{4}}E_T(2,3,4) + P_{13}^{\bar{4}}E_T(3,4) + 1. \end{aligned} \tag{3.6}$$

We can compute the shipping time from the first warehouse in a similar fashion:

$$\begin{aligned} E_T(2,3,4) &= (1 - P_{23}^{\bar{4}} - P_{24})(E_T(2,3,4) + 1) \\ &+\ P_{23}^{\bar{4}}(E_T(3,4) + 1) + P_{24} \\ (P_{23}^{\bar{4}} + P_{24})E_T(2,3,4) &= P_{23}^{\bar{4}}E_T(3,4) + 1. \end{aligned} \tag{3.7}$$

Before we prove Theorem 2, we make a few observations.

Lemma 7. *If $E_T(1,2,3,4) < E_T(1,4)$ then $E_T(2,3,4) < E_T(1,4)$ or $E_T(3,4) < E_T(1,4)$.*

In other words, if the shipping time with two warehouses is smaller than the nominal shipping time, then at least one of the shipping times from the warehouses is smaller than the nominal shipping time.

Proof. We prove Lemma 7 by contrapositive and assume that $E_T(2,3,4) > E_T(1,4)$ and $E_T(3,4) > E_T(1,4)$. The proof shows that the shipping time with two warehouses under the made assumption is always bigger than the nominal shipping time.

$$\begin{aligned}(P_{12}^{\bar{3}\bar{4}} + P_{13}^{\bar{4}} + P_{14})E_T(1,2,3,4) &= P_{12}^{\bar{3}\bar{4}}E_T(2,3,4) + P_{13}^{\bar{4}}E_T(3,4) + 1 && \text{from (3.6)}\\ &> P_{12}^{\bar{3}\bar{4}}E_T(1,4) + P_{13}^{\bar{4}}E_T(1,4) + 1\\ &= (P_{12}^{\bar{3}\bar{4}} + P_{13}^{\bar{4}} + P_{14})E_T(1,4)\end{aligned}$$

The inequality follows from the assumptions. The proof is completed by the fact that $P_{14}E_T(1,4) = 1$. □

Lemma 8. $E_T(1,2,3,4) < E_T(1,4)$ *implies* $E_T(1,2,4) < E_T(1,4)$ *or* $E_T(1,3,4) < E_T(1,4)$.

In other words, if the shipping time with two warehouses is smaller than the nominal shipping time, then at least one of the warehouses belongs to the set W_G.

Proof. We prove Lemma 8 by contrapositive and assume that none of the warehouses is placed at a good warehouse location. The proof shows that the shipping time with two warehouses under the made assumption is always bigger than the nominal shipping time.

$$\begin{aligned}\text{we assume}\quad & E_T(1,2,4) > E_T(1,4) \quad \text{and} && (3.8)\\ & E_T(1,3,4) > E_T(1,4) && (3.9)\\ \text{which will show}\quad & E_T(2,3,4) > E_T(1,4) \quad \text{and} && (3.10)\\ & E_T(3,4) > E_T(1,4) && (3.11)\\ \text{which by Lemma 7 implies}\quad & E_T(1,2,3,4) > E_T(1,4)\end{aligned}$$

Immediately, (3.11) follows from (3.9) by (3.5). Now we show that (3.10) is true under the made assumptions.

$$\begin{aligned}
(P_{23}^{\bar{4}} + P_{24})E_T(2,3,4) &= P_{23}^{\bar{4}}E_T(3,4) + 1 && \text{from (3.7)}\\
&> P_{23}^{\bar{4}}E_T(1,4) + 1\\
&= P_{23}^{\bar{4}}E_T(1,4) + P_{24}E_T(2,4)\\
&> P_{23}^{\bar{4}}E_T(1,4) + P_{24}E_T(1,4)\\
&= (P_{23}^{\bar{4}} + P_{24})E_T(1,4)
\end{aligned}$$

The first inequality follows from (3.11) and the second inequality follows from the fact that $E_T(2,4) > E_T(1,4)$, which is true by (3.8) in conjunction with (3.5). □

Theorem 2. *The optimal shipping scheme for a network with two warehouses has the following properties:*

- $E_T(1,3,4) < E_T(1,4)$ *and*
- $P_{24} < P_{34}$.

Proof. We start and prove $P_{24} < P_{34}$ by contrapositive. We assume that $P_{24} > P_{34}$ and show that the shipping time $E_T(1,2,3,4)$ is not minimal, and one can find a shipping scheme with smaller shipping time. The assumption implies inequalities (3.12)–(3.13) by Lemma 6 and inequality (3.14) because of the geometric behavior of the shipping time with no warehouses between source and sink.

$$E_T(2,3,4) > E_T(2,4) \tag{3.12}$$

$$E_T(3,4) > E_T(3,2,4) \tag{3.13}$$

$$E_T(3,4) > E_T(2,4) \tag{3.14}$$

$$\begin{aligned}
(P_{12}^{\bar{3}\bar{4}} + P_{13}^{\bar{4}} + P_{14})E_T(1,2,3,4) &= P_{12}^{\bar{3}\bar{4}}E_T(2,3,4) + P_{13}^{\bar{4}}E_T(3,4) + 1 && \text{from (3.6)}\\
&> P_{12}^{\bar{3}\bar{4}}E_T(2,4) + P_{13}^{\bar{4}}E_T(3,4) + 1 && \text{by (3.12)}\\
&> P_{12}^{\bar{4}}E_T(2,4) + P_{13}^{\bar{2}\bar{4}}E_T(3,4) + 1 && (3.15)\\
&> P_{12}^{\bar{4}}E_T(2,4) + P_{13}^{\bar{2}\bar{4}}E_T(3,2,4) + 1 && \text{by (3.13)}\\
&= (P_{13}^{\bar{2}\bar{4}} + P_{12}^{\bar{4}} + P_{14})E_T(1,3,2,4)
\end{aligned}$$

$$E_T(1,2,3,4) > E_T(1,3,2,4) \tag{3.16}$$

To show that inequalities (3.15) and (3.16) hold we make the following observations:

$$P_{13}^{\bar{2}\bar{4}} \leq P_{13}^{\bar{4}} \quad \text{and} \quad P_{12}^{\bar{4}} \geq P_{12}^{\bar{3}\bar{4}}$$

and

$$\begin{aligned} P_{1(2,3)}^{\bar{4}} &= P_{1(3,2)}^{\bar{4}} && (3.17) \\ P_{1(2,3)}^{\bar{4}} - P_{13}^{\bar{4}} + P_{13}^{\bar{4}} &= P_{1(3,2)}^{\bar{4}} - P_{12}^{\bar{4}} + P_{12}^{\bar{4}} \\ P_{12}^{\bar{3}\bar{4}} + P_{13}^{\bar{4}} &= P_{13}^{\bar{2}\bar{4}} + P_{12}^{\bar{4}} && (3.18) \end{aligned}$$

In (3.17), we start with the probability that the operator can ship from the source to one of the warehouses, and the order in a disjunction does not matter. We add zero on both sides of the equality, and collect terms using the technique expressed by the Venn Diagram in Figure 3.4.

To obtain (3.15), we take some probability mass that weights $E_T(3,4)$ and increase the weight for $E_T(2,4)$, leaving the sum of the probabilities constant (3.18). The inequality still holds because $E_T(3,4) > E_T(2,4)$, by (3.14). In the last step of the argument, (3.16), we remove the common factor from both sides of the equation.

The first part of Theorem 2, $E_T(1,3,4) < E_T(1,4)$, is proven as follows. Lemma 8 shows that at least one of the warehouse locations belongs to the set of good warehouse locations,

$$E_T(1,2,4) < E_T(1,4) \quad \text{or} \quad E_T(1,3,4) < E_T(1,4)$$

which implies by Lemma 6

$$P_{24} > P_{14} \quad \text{or} \quad P_{34} > P_{14}.$$

In conjunction with $P_{24} < P_{34}$ it follows that $P_{34} > P_{14}$ and therefore, $E_T(1,3,4) < E_T(1,4)$, which competes the proof. □

To illustrate the impact we use the Air Force network, which has 93 possible warehouse locations, and consider adding warehouses in three different shipping scenarios. All three scenarios have the same destination, Bosnia, but differ on the cargo origin: Dover, March, or Mayport. For all three scenarios we calculate the shipping time for each possible warehouse added to the network. After that calculation, including necessary probability estimations, we identify the set of good warehouse locations, W_G, in each scenario and apply Theorem 2 in two stages. In stage one, we consider $w \in W_G$ and $v \in N \setminus \{s, w, t\}$, and in stage two we only consider warehouse combinations from the first stage that maintain $P_{vt} < P_{wt}$. The number of resulting warehouse combinations for the Air Force network is shown in Figure 3.7.

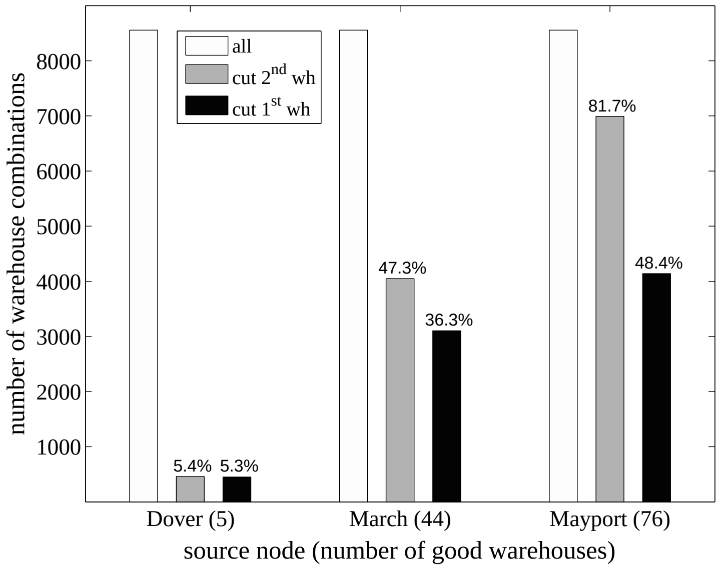


Figure 3.7: **Use of theorem 2** - We consider the Air Force network with Bosnia as sink and three different locations as source. The numbers next to the source location identify the size of W_G. The figure shows the number of possible warehouse combinations. The first bar in each group represents the number of all possible combinations. The second bar (cut 2$^{\text{nd}}$ wh) shows the number of combinations if the warehouse w is chosen to be out of W_G and the third bar (cut 1$^{\text{st}}$ wh) represents all combinations that maintain the inequality $P_{vt} < P_{wt}$. In all three scenarios, we see at least a 50% reduction in the number of possible solutions by applying Theorem 2, and in scenario 1, we see a 95% reduction.

3.4.4 Bounding Shipping Time

Adding more warehouses to a stochastic network has the potential to increase the network's performance, but one expects a diminishing return on investment by adding more and more warehouses. A bound on the best possible shipping time based on an infinite budget for placing warehouses allows us to calculate the gap to the current optimal solution. That gap may indicate that it is not worth adding another warehouse before executing the time consuming simulation. Computing the hitting time between s and t for the model in Figure 3.8 provides such a bound. After finding the best single warehouse, we have estimates for all the probabilities we need to evaluate (3.20).

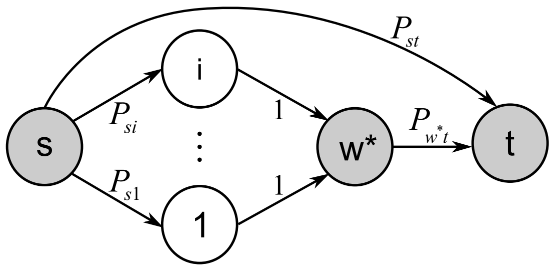


Figure 3.8: **Model for lower bound** - We define w^* as the location with shortest shipping time $E_T(w^*, t)$. That location is identified while finding the best single warehouse location as well as the probability P_{st}. The outgoing arcs from the source can be replaced by a single arc, with assigned probability that at least one of the outgoing arcs is present. We call that probability P_{sout}. The shipping time from s to t for this model is a lower bound on the shipping time, regardless of the number of warehouses one would add to the network.

We denote by $p_{(s,v)}$ the probability that the arc (s, v) is available, and by P_{sout} the probability that at least one of the source's outgoing arcs is present.

$$P_{\text{sout}} = 1 - \prod_{\substack{v: \\ (s,v)\in A}} (1 - p_{(s,v)})$$

We identify the node w^*, which is the node with highest P_{wt}, after running the simulation to find the best single warehouse,

$$w^* = \underset{w\in N\setminus\{t\}}{\arg\max}\{P_{wt}\} \tag{3.19}$$

and calculate the lower bound, LB, for the shipping time from s to t as:

$$LB = \frac{1 + \dfrac{P_{\text{sout}}}{P_{w^*t}}}{P_{st} + P_{\text{sout}}}. \tag{3.20}$$

The lower bound equation (3.20) is similar to the equation for calculating the shipping time with one warehouse. First, we have a closer look at this equation, in particular, at the behavior if we change $P_{12}^{\bar{3}}$.

$$E_T(1,2,3) = \frac{1 + \frac{P_{12}^{\bar{3}}}{P_{23}}}{P_{13} + P_{12}^{\bar{3}}}$$

$$\frac{\partial E_T(1,2,3)}{\partial P_{12}^{\bar{3}}} = \frac{\frac{P_{13}}{P_{23}} - 1}{(P_{13} + P_{12}^{\bar{3}})^2} \begin{cases} > 0, & \text{if } P_{13} > P_{23} \\ < 0, & \text{if } P_{13} < P_{23} \end{cases}$$

$$\Rightarrow E_T(1,2,3) \quad \text{is} \quad \begin{cases} \text{monotone increasing in } P_{12}^{\bar{3}} \text{ if } P_{13} > P_{23} \\ \text{monotone decreasing in } P_{12}^{\bar{3}} \text{ if } P_{13} < P_{23} \end{cases} \tag{3.21}$$

Before we prove that the lower bound holds, we make a few observations. We define the set of all possible shipping schemes, Δ_{xt}, for shipping from any node, $x \in N \setminus \{t\}$, of the network to the sink node t, regardless the number of warehouses.

Lemma 9. $E_T(w^*, t) = \min_{\delta \in \Delta_{xt}} E(\delta)$

Verbalized, regardless of the number of warehouses, there is no shipping scheme with an expected shipping time that is lower than $E_T(w^*, t)$.

Proof. By Lemma 6 and (3.19), the shipping time $E_T(w^*, t)$ cannot be improved by adding warehouses between w^* and t. In addition, one can use inductive arguments to show that no matter how many warehouses we introduce, there is no node $x \neq w^*$, such that $E_T(x, \dots, t) < E_T(w^*, t)$. The trivial case with no warehouses at all will have a longer expected shipping time, simply by our choice of w^*. For the network with one warehouse and an arbitrary start node, we formulate the shipping time according to the Markov chain model.

$$\begin{aligned} E_T(x,w,t) &= (1 - P_{xw}^{\bar{t}} - P_{xt})(E_T(x,w,t) + 1) \\ &\quad + P_{xw}^{\bar{t}}(E_T(w,t) + 1) + P_{xt} \\ (P_{xw}^{\bar{t}} + P_{xt})E_T(x,w,t) &= P_{xw}^{\bar{t}} E_T(w,t) + 1 \\ &= P_{xw}^{\bar{t}} E_T(w,t) + P_{xt} E_T(x,t) \\ &> (P_{xw}^{\bar{t}} + P_{xt}) E_T(w^*, t) \end{aligned}$$

The inequality follows from (3.19) and shows that no matter what source node, x, and warehouse location, w, we pick, $E_T(x, w, t) \geq E_T(w^*, t)$. One can proceed in the same manner and add one warehouse after another. With $k+1$ warehouses, we use the fact that for all cases with $0, 1, \ldots, k$ warehouses the shipping time from any node to t is never smaller than $E_T(w^*, t)$ to complete the proof. $\square$

The lower bound holds for an arbitrary number, n, of warehouses in the network. In an abstract network representation we have the nodes $1, \ldots, n{+}2$, where node 1 is the source and node $n{+}2$ represents the sink. According to our notation, the probability of shipping from the source to the warehouse i, is $P_{1i}^{\overline{i+1}\ldots\overline{n+2}}$. We denote all events contributing to this probability by $\mathcal{E}_i$, which are all the network states including a path from s to the warehouse i, but no path from s to node j, where $j > i$. Notice that the event sets for all warehouses are pairwise disjoint.

$$\mathcal{E}_i \cap \mathcal{E}_j = \emptyset, \qquad 2 \leq i \leq n+1, \quad 2 \leq j \leq n+1, \quad i \neq j$$

The disjointness of the success events allows us to add the probabilities without over counting. The resulting sum is the probability that one can ship from the source to one of the n warehouses. And, no matter what warehouse we are shipping to, at least one of the outgoing arcs of the source needs to be present in the network. Thus,

$$P_{\text{sout}} \geq \sum_{i=2}^{n+1} P_{1i}^{\overline{i+1}\ldots\overline{n+2}}. \tag{3.22}$$

We are going to substitute one term for another later.

Now we prove that LB is indeed a lower bound that can never be undercut, regardless of the number of warehouses added to the network.

Proof. We consider a shipping scheme with an arbitrary number of warehouses, n, and find its expected shipping time (3.23).

$$\begin{aligned} E_T(1,\ldots,n{+}2) &= \left(1-\sum_{i=2}^{n+1} P_{1i}^{\overline{i+1}\ldots\overline{n+2}} - P_{1n+2}\right)(E_T(1,\ldots,n{+}2)+1) \\ &\quad + \sum_{i=2}^{n+1}\left[P_{1i}^{\overline{i+1}\ldots\overline{n+2}}(E_T(i,\ldots,n{+}2)+1)\right] + P_{1n+2} \qquad (3.23) \\ &= \frac{1+\sum_{i=2}^{n+1}\left[P_{1i}^{\overline{i+1}\ldots\overline{n+2}}E_T(i,\ldots,n{+}2)\right]}{\sum_{i=2}^{n+1}P_{1i}^{\overline{i+1}\ldots\overline{n+2}}+P_{1n+2}} \qquad (3.24) \\ &> \frac{1+\sum_{i=2}^{n+1}P_{1i}^{\overline{i+1}\ldots\overline{n+2}}E_T(w^*,t)}{\sum_{i=2}^{n+1}P_{1i}^{\overline{i+1}\ldots\overline{n+2}}+P_{1n+2}} \\ &> \frac{1+P_{1\mathrm{out}}E_T(w^*,t)}{P_{1\mathrm{out}}+P_{1n+2}} \end{aligned}$$

The shipping time (3.23) consists of three parts weighted by the corresponding probabilities: (1) the shipping time if the cargo stays at the source, (2) the shipping time if the cargo moves to any warehouse, and (3) the shipping time if the cargo moves to the sink. We collect terms and divide by the factor that would appear on the left hand side (3.24). The first inequality follows from Lemma 9. The probability P_{w^*t} is greater than P_{1n+2} by (3.19), and therefore the equation is monotone decreasing with increasing $\sum_{i=2}^{n+1} P_{1i}^{\overline{i+1}\ldots\overline{n+2}}$ by (3.21). The probability $P_{1\mathrm{out}}$ is an overestimate for $\sum_{i=2}^{n+1} P_{1i}^{\overline{i+1}\ldots\overline{n+2}}$ by (3.22), and substituted in the last inequality gives the lower bound equation, which completes the proof. □

To show an example for the lower bound, and the gaps between solutions and bound, we use the Air Force data and pick Dover, March, and Mayport as source nodes, and Bosnia as the sink node. In Figure 3.9, we plot the nominal shipping time, the optimal shipping time for one and two warehouses as well as the lower bound.

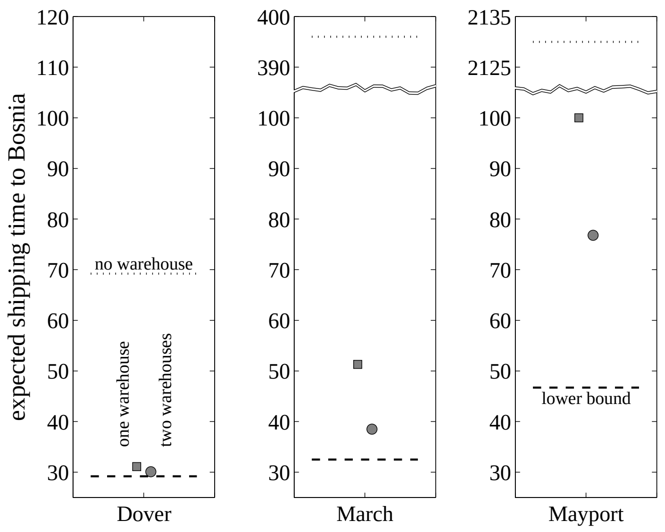


Figure 3.9: **Shipping times and bounds to Bosnia** - We used the Air Force data and picked three different source nodes and Bosnia as sink node. The plot shows the nominal shipping times, the optimal shipping times with one and two warehouses as well as the lower bound. Notice the significant drop in shipping time due to the first warehouse as compared to the marginal decrease in shipping time with the second warehouse. Because the two-warehouse solution is so close to the lower bound, it says that adding additional warehouses to this network may not be worth the investment.

3.5 Simulation Techniques

3.5.1 Network Sampling

The network G is held in computer memory as a list of nodes and a list of arcs. To sample the stochastic network one needs to iterate over the arcs, and either kept it or remove it according to the current arc state. The state for arc a is a Bernoulli random variable, with probability of success p_a. The modified arc list represents the current state of the network, for which one has to identify whether s and t are connected. We use a breadth-first search (BFS) procedure for identification. BFS takes a network and a start node as input, and returns the set of all nodes, R, reachable from the start node. If $t \in R = \text{BFS}(G, s)$ the sample is a success, and a counter associated with the estimate for P_{st} is incremented.

Building the arc list repetitively is a costly technique and can be avoided. We keep G with all arcs turned on, and introduce an indicator vector I which is the current arc state (active, inactive) for each arc. To sample the network, we update each entry of I

according to the outcome of a Bernoulli random experiment, as shown in Algorithm 3.1. To identify whether the sample is a success, we need both G and I. We customize a BFS procedure as shown in Algorithm 3.2, line 8, to account for the indicator variables.

```
 1: **procedure** SAMPLENETWORK($G = (N, A), P$)
 2:     **for each** $a \in A$ **do**
 3:         $I(a) = \text{Bernoulli}(P(a))$
 4:     **end for**
 5:     return $I$
 6: **end procedure**
```

Algorithm 3.1: **Sampling procedure** - The sampling procedure iterates over the list of arcs and performs a Bernoulli random trial for each arc, with associated probability $P(a) = p_a$. The results are stored as the indicator vector, I, which represents the current network state.

```
 1: **procedure** BFS($G, u, I$)
 2:     initialize queue $Q$
 3:     $Q$.push($u$)
 4:     $R \leftarrow \emptyset$          list of visited nodes
 5:     **while** $Q$ is not empty **do**
 6:         $v \leftarrow Q$.pop(0)
 7:         **for each** $n \in N$ such that $a = (v, n) \in A$ and $n \notin R$ **do**
 8:             **if** $I(a)$ **then**
 9:                 $Q$.push($n$)
10:             **end if**
11:         **end for**
12:         $R$.add($v$)
13:     **end while**
14:     return $R$
15: **end procedure**
```

Algorithm 3.2: **Customized BFS** - The customized BFS procedure uses the indicator vector and considers only arcs that are present in the current network state.

A simulation sequence samples each arc, followed by a call of the BFS procedure, which traverses the network using present outgoing arcs of reachable nodes. Outgoing arcs of unreachable nodes are not used in the procedure, nor arcs going from the node, currently processed inside BFS, to a node already marked as visited. Sampling such arcs is unnecessary. Therefore, we incorporate the arc sampling into a new BFS$^*(G, s, P)$, and instead of I, we provide P as input. Pseudo code for BFS* is shown as Algorithm 3.3.

```
1: procedure $BFS^*(G, u, P)$
7:     for each $n \in N$ such that $a = (v, n) \in A$ and $n \notin R$ do
8:         if Bernoulli($P(a)$) then
9:             $Q$.push($n$)
10:        end if
11:    end for
15: end procedure
```

Algorithm 3.3: **BFS***- The procedure BFS* only samples the arcs needed to proceed traversing the network. The pseudo code for BFS* shows the difference to Algorithm 3.2, i.e., the lines that have to be swapped out in the customized BFS procedure.

source	Dover	March	Mayport
d_{out}	39	13	7
p_{out}	0.965	0.244	0.056
p_{st}	0.01453	0.00252	0.00047
fraction of arcs sampled by BFS* [%]	14.3	3.7	1.1
runtime BFS* [sec]	772	205	69
runtime for sampling all arcs plus BFS [sec]	2878	2444	2342

Table 3.1: **BFS* gain** - We use the Air Force network (99 nodes, 1103 arcs) and Bosnia as sink node. We perform three million iterations for each method. First, we sample the network and execute one BFS call, according to Algorithm 3.2. The resulting runtime is shown in the last row. Second, we call BFS*. The number of outgoing arcs for a sink node is shown as d_{out}, and p_{out} is the probability that at least one of the outgoing arcs is present. The numbers show the significant saving in runtime due to the BFS* procedure, up to a factor of more than 30.

The computational saving using BFS* depends on the arc probabilities, as well as on the start node. To show this, we use the Air Force network and arbitrarily pick three different start nodes and a sink node. First, we use the sampling technique that builds the I vector and identify if each sample is successful. The number of arcs sampled, n_a, is the number of arcs in the network times the number of samples drawn. For comparison, we clock the runtime to draw and identify three million samples. Second, we use BFS*, count the number of arcs sampled and clock the runtime. Runtimes and arcs sampled as the fraction of n_a are shown in Table 3.1. In addition, we show the number of outgoing arcs, d_{out}, and the probability that at least one of the outgoing arcs is present, p_{out}, for each source node.

3.5.2 More Efficient Simulation

One can argue against the use of BFS* to identify whether the network sample contains a path from source to sink. A depth-first search (DFS) procedure can potentially be the better choice, where better means less work. In this paragraph, we propose an efficient simulation sequence that supports the use of a BFS algorithm.

We first consider adding a single warehouse and remind the reader, that we have to estimate the probabilities P_{st} and $P_{sw}^{\bar{t}}$, $\forall w \in W$. One single BFS$^*(G, s, P)$ call can be used to update the counts associated with each of the probabilities. In addition we need to estimate the probabilities P_{wt}, $\forall w \in W$. Now, we have a changing start node, w, but a constant sink node. Instead of calling BFS* for each $w \in W$, we call one BFS$^*(G_r, t, P)$ on the reversed network G_r, that is, all arcs from the original network point in the opposite direction. We update the count associated with the probability P_{wt} if $w \in R = $ BFS$^*(G_r, t, P)$. Counts are divided by the number of network samples to turn them into probabilities at the end of the simulation sequence. We call BFS* two times the number of samples required to keep the CI in the desired limits, calculate the shipping time for each warehouse location, and find the best warehouse location for adding a single warehouse. With respect to the three cost affecting factors for the Monte Carlo experiment, we blend sampling and identifying of successful samples into each other, and we shrink the effort of drawing a sample by sampling relevant arcs only. The time saved can be used to draw more samples, which affects the sampling error.

We solve the WLNR problem with one warehouse for the Air Force network in about 1500 seconds, calling BFS$^*(G, s, P)$ as well as BFS$^*(G_r, t, P)$ five million times. We use Dover as source and Bosnia as sink for this particular runtime example.

Solving the WLNR problem considering two warehouses, we can reuse the probabilities estimated during solving the WLNR problem with one warehouse, but have to estimate the probabilities $P_{sv}^{\overline{w}\bar{t}}$ and $P_{vw}^{\bar{t}}$, $\forall v, w \in W$, as already mentioned. We can estimate the probabilities $P_{sv}^{\overline{w}\bar{t}}$, $\forall v, w \in W$ in a similar way we estimated $P_{sw}^{\bar{t}}$. One BFS* call is sufficient to update all the associated counts. To estimate the probabilities $P_{vw}^{\bar{t}}$, $\forall v, w \in W$ we have to call BFS* multiple times, because there is no common source nor common sink to identify. We notice that the number of good warehouse locations $|W_G|$ is potentially significantly smaller than the warehouse locations that could serve as warehouse v. Thus,

```
1: $W$          set of possible warehouse locations
2: $R \leftarrow \emptyset$          set of reachable nodes
3: $count_{st} \leftarrow 0$
4: $count^{\bar{t}}_{sw} \leftarrow 0,\ count_{wt} \leftarrow 0,\ \forall w \in W$
5: for $i = 0$ to $numSamples$ do
6:     $R \leftarrow \text{BFS}^*(G, s, P)$
7:     if $t \in R$ then
8:         $count_{st}$ += 1
9:     else
10:        for each $n \in R$ do
11:            $count^{\bar{t}}_{sn}$ += 1
12:        end for
13:    end if
14:    $R \leftarrow \text{BFS}^*(G_r, t, P)$
15:    for each $n \in R$ and $n \neq s$ do
16:        $count_{nt}$ += 1
17:    end for
18: end for
```

Algorithm 3.4: **Simulation for single warehouse probabilities** - For each sample, this algorithm performs two BFS* procedures, the first on the original network and the second on the reversed network. Counters associated with the reachable nodes are updated after each call. The conditional update in lines 7-13 maintains the definition of the probabilities $P^{\bar{t}}_{sw}$.

calling BFS$^*(G_r, w, P)$ is potentially the way of less work, but it does not return the desired probabilities. We use the technique pictured by the Venn diagram in Figure 3.4. We add the permanent arc (w, t), and increment the count associated with the probability $P_{v(w,t)}$, if $v \in R = \text{BFS}^*(G_r, \{w, t\}, P)$. Adding the permanent arc (w, t) is equivalent to initially pushing w and t onto the queue utilized by BFS*. From the resulting probabilities one has to subtract the corresponding probabilities P_{vt}, which we estimate while solving the WLNR problem for one warehouse.

We solve the WLNR problem with two warehouses for the Air Force network and a given source sink pair in about two hours by the described simulation sequence with five million replications. We use Dover as source and Bosnia as sink for this particular runtime example. For comparison, we run the same example, ignoring the results from Theorem 2, which enlarges the runtime to nine hours due to calculating the shipping times for all warehouse combinations, including the runtime to estimate the relevant probabilities.

3.6 Solution for Dozier's Transportation Network

To find the optimal warehouse locations for a network with several sources and multiple sinks, one has to consider each source-sink pair and find the corresponding shipping time for each combination of two warehouse locations. The shipping times can be weighted according to source sink relevance or treated equally to find the best warehouse locations.

Dozier's data set does not provide information about the importance of source, sink, or source sink combination. As sink locations, we illustratively pick the four locations that host the main airlift components for the U.S. Air Force: Charleston, Dover, McChord, and Travis. We assume that humanitarian assistance cargo is equally likely generated in one of the four locations and equally likely to be shipped to one of 13 sink locations. Three of the original 17 sink locations are not reachable in the AF Air network. Sink location Croatia, with its very large nominal shipping time, dictates the location for the best warehouse locations, and is therefore removed from considerations. We conduct five million replications for estimating the reliabilities.

No matter the origin of the humanitarian assistance cargo, the best location for one warehouse is Ramstein. Placing one warehouse at Sigonella and one at Ramstein is the optimal solution for placing two warehouses. Here Sigonella serves as the first warehouse and Ramstein as the second warehouse, i.e., shipping from Ramstein to Sigonella is not considered in the optimal shipping policy. The specific shipping times can be read from the plots in Figure 3.10–Figure 3.13. The solution shows that adding a second warehouse adds only a marginal benefit and the Sigonella is only used for shipping to three of the 13 sink locations. Here, the solution is nested, meaning the best location for a single warehouse is also the best location for the second warehouse in the two-warehouse solution. We do not always see nested solutions, and a counterexample can be easily created.

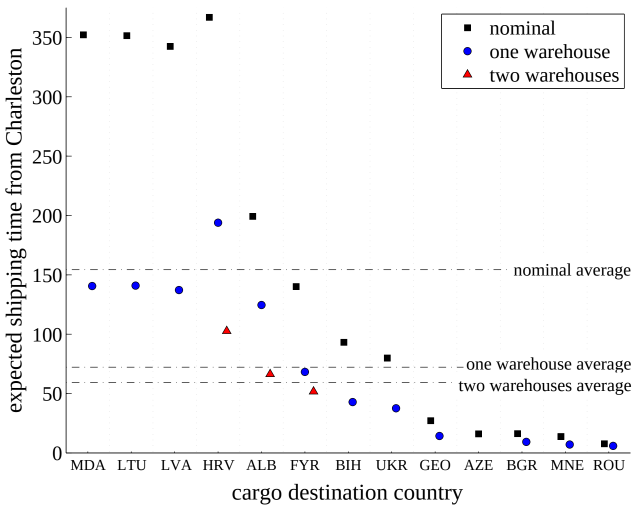


Figure 3.10: **Shipping times from Charleston** - A single warehouse at Ramstein or one at Sigonella and one at Ramstein result in the lowest average shipping times for one and two warehouses, respectively. The plot shows the nominal shipping times, the shipping times with only Ramstein in place, and the shipping times with both warehouses. A missing data point, e.g., for MDA, indicates that the optimal shipping policy never considers both warehouses. Here, one would never ship to Sigonella and rather wait at the source until there is a path to MDA or to Ramstein available.

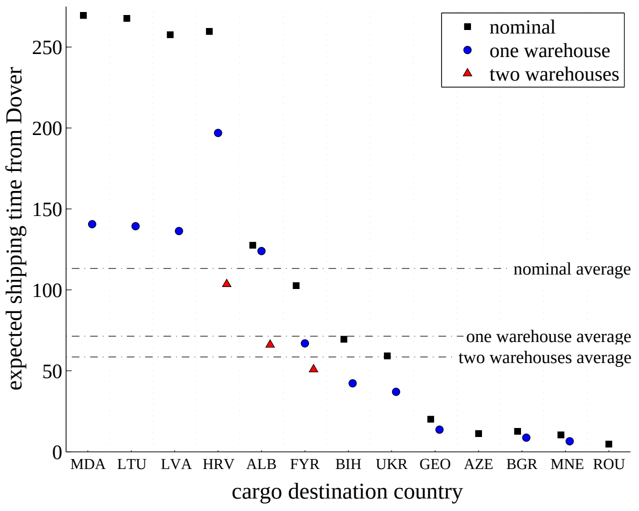


Figure 3.11: **Shipping times from Dover** - A single warehouse at Ramstein or one at Sigonella and one at Ramstein result in the lowest average shipping times for one and two warehouses, respectively. The plot shows the nominal shipping times, the shipping times with only Ramstein in place, and the shipping times with both warehouses. A missing data point, e.g., for MDA, indicates that the optimal shipping policy never considers both warehouses. Here, one would never ship to Sigonella and rather wait at the source until there is a path to MDA or to Ramstein available. In the case that AZE is the sink location, shipping to a warehouse would increase the shipping time, and therefore, the optimal shipping policy never ships to a warehouse.

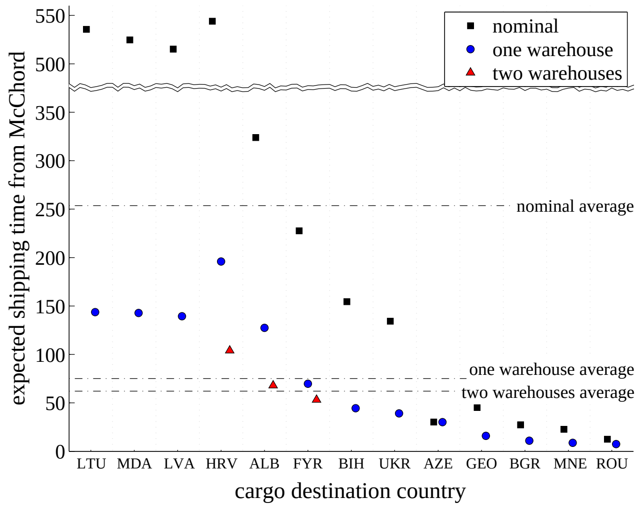


Figure 3.12: **Shipping times from McChord** - A single warehouse at Ramstein or one at Sigonella and one at Ramstein result in the lowest average shipping times for one and two warehouses, respectively. The plot shows the nominal shipping times, the shipping times with only Ramstein in place, and the shipping times with both warehouses. A missing data point, e.g., for LTU, indicates that the optimal shipping policy never considers both warehouses. Here, one would never ship to Sigonella and rather wait at the source until there is a path to LTU or to Ramstein available.

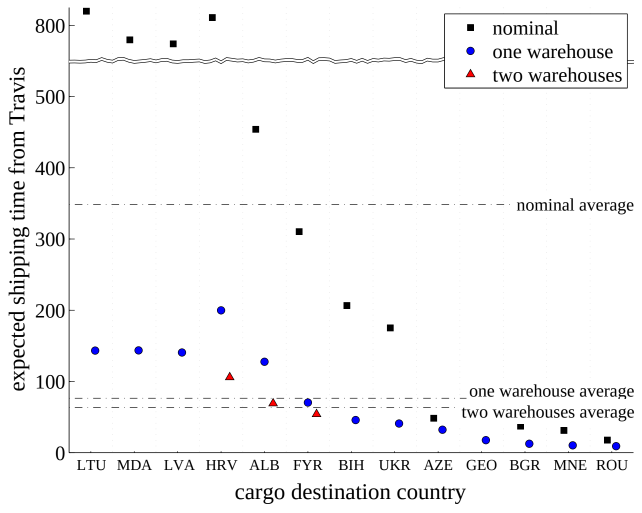


Figure 3.13: **Shipping times from Travis** - A single warehouse at Ramstein or one at Sigonella and one at Ramstein result in the lowest average shipping times for one and two warehouses, respectively. The plot shows the nominal shipping times, the shipping times with only Ramstein in place, and the shipping times with both warehouses. A missing data point, e.g., for LTU, indicates that the optimal shipping policy never considers both warehouses. Here, one would never ship to Sigonella and rather wait at the source until there is a path to LTU or to Ramstein available.

3.7 Conclusion

The Warehouse Location Network Reliability problem (WLNR) considers a stochastic logistics network, where arcs can fail randomly and independently. In this network, one wishes to ship, but shipment must be delayed until the network provides a complete path from the cargo's source to sink. Introducing storage capacity to some of the nodes enables partway shipment that can decrease the shipping time. WLNR identifies the storage locations that optimize shipping time and defines the associated optimal shipping policy.

WLNR can be formulated as a Markov Decision Process with Design (MDPD), which solution provides both the optimal warehouse location and the optimal shipping policy. Such an MDPD is huge, with network size growing state space, and is therefore intractable for large networks. However, the results from the MDPD reveal the structure for the optimal shipping policy. That policy ships to the reachable node with storage capacity (a warehouse or sink), from which one expects the smallest shipping time to the sink. Therefore, between any two nodes with storage capacity, only one shipping direction appears in the optimal shipping policy. We pre-define such an optimal shipping policy. For fixed warehouse locations, the MDPD with pre-defined shipping policy behaves like a Markov chain. We use such a Markov chain to compute the expected shipping time for one or two warehouses. The resulting equations involve several s-t-reliabilities. Computing s-t-reliability is known to be #P-complete and impracticable for large networks.

We provide methods to find the optimal warehouse locations for one and two warehouses. Estimating reliability by Monte Carlo simulation, which is very time consuming, lies at the core of the methods to solve WLNR. We show theoretical results that allow the elimination of suboptimal solutions from the solution space, resulting in a significant saving of computational effort. We create a case study from the DOD's humanitarian assistance transportation network with 95 nodes and 1096 arcs. There are more than 8000 possible ways to place two warehouses in that network. We identify up to 95% of these combinations as being suboptimal without computing the associated shipping times, which saves on simulation effort. We achieve further savings of simulation time by customizing the sampling method. We use a technique that only samples arcs as needed while searching for a path from s to t. In the case study, we sample less than 20% of all arcs when applying this technique. We also provide a theoretical bound on the best possible expected waiting time. The bound shows in the case study example that adding more than two warehouses improves results only fractionally.

In our model we only consider cargo with a fixed size of one unit, and our solution suggests where to introduce warehouses but does not address the size of the warehouse. Future models that consider cargo and space available of different sizes could, in addition to selecting warehouse locations, also find required warehouse sizes. Dozier's data set is not qualified for such an analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4: Conclusion and Future Work

This dissertation considers two network design problems. We define the Path-Portfolio-Selection problem and the Sub-Network-Selection problem to address designing transportation networks that are resilient to attacks. Random failures are considered in the Warehouse Location Network Reliability problem, where we add storage capacity to a stochastic logistics network to gain reliability.

The Path-Portfolio-m-n problem (PPS-m-n) and the Sub-Network-Selection-m-n problem (SNS-m-n) are three-stage sequential games, leading to tri-level optimization problems, where the inner level point in the opposite direction from the outer level, i.e. min-max-min. The dissertation shows how these problems are equivalent to single-level models that are theoretically solvable through integer programming methods. The number of variables and constraints in the resulting models, however, grows exponentially with size of the network, which renders the models intractable for large networks. We find a practicable solution method for large networks that involves both row and column generation. Savings in solution time are obtained by adding supervalid inequalities to the model, and removing superfluous parts from the network. With the proposed methods, we are able to solve PPS-m-n on a road network of Germany with 2,971 nodes and 8,824 arcs in about two hours. For a U.S. road network with 47,941 nodes and 124,414 arcs we find a feasible solution for PPS-m-n with an optimality gap of 0.07% in less than an hour.

We further explore several variations of PPS-m-n and SNS-m-n. The probabilistic versions Pr-PPS-m-n and Pr-SNS-m-n incorporate attack probabilities to the models, accounting for uncertainty about the attacker's capability. The uncertainty about the number of attacked arcs requires additional constraints to identify the defender's response to the attack. Similar constraints are necessary to identify the defender's response in the general cases D-PPS-m-n and D-SNS-m-n, where an attacked arc is considered to be lengthened rather than destroyed. Here, the response to an attack could be an interdicted path, which makes the problem difficult to solve. In C-PPS-m-n and C-SNS-m-n the attacker is allowed to "smear" the attack budget continuously across the arcs in the network. The continuous relaxation of the attack variables enables a two-step reformulation of the inner optimization problem, leading to a single -level optimization model. We further generalize SNS-m-n to an arbitrary minimum cost flow problem, showing that essentially all network models can be incorporated into the framework we create.

For future work, we point to the possibility of extending PPS-m-n to other network problems in a similar fashion as SNS-m-n. An attacked arc in PPS-m-n is destroyed, and any path in the portfolio containing an attacked arc is not usable for operation. A shipping plan in the sense of minimum cost flow problems typically uses more than one path from source to sink. Therefore, destroying arcs in a pre-planned shipping plan does not necessarily render the shipping plan infeasible or suboptimal. The possibility that interdicted shipping plans may be optimal leads to an D-PPS-m-n like model, which is difficult to solve.

Furthermore, there is potential to improve the models and reduce solution times. Especially for the general case D-PPS-m-n, there is potential for savings by including supervalid inequalities. Methods to remove superfluous parts of the network, as demonstrated for PPS-m-n, can potentially be extended to D-PPS-m-n and serve as the heart of a customized branch-and-bound algorithm. While it progresses, branch and bound encounters feasible solutions that are used to obtain bounds on the objective function value. Typically, such feasible solutions are not returned to the user. One could, however, extract an upper bound for the longest path in the portfolio from each of the suboptimal feasible solutions, shrink the network appropriately, and continue the B&B process on the smaller network.

The core of our solution methods for PPS-m-n and SNS-m-n is the reformulation of the otherwise tri-level optimization problem as a single-level optimization problem, which raises the question whether this is a problem-specific characteristic, or if it applies to arbitrary tri-level optimization problems. In our models, we explicitly list all possible attacks and link the worst of these attacks to the objective function, which to some degree represents the second stage of the optimization problem. The resulting single-level model is exponentially large, but a row-and-column generation procedure turns out to be a practicable solution approach. Listing the second stage in a similar way may be a good starting point to tackle an arbitrary tri-level optimization problem.

The Warehouse Location Network Reliability problem (WLNR) considers a stochastic logistics network, where arcs can fail randomly and independently. Because of the absence of storage capacity, cargo shipping must be delayed until the network provides a complete path from source to sink. Introducing storage capacity enables partway shipment, which can decrease the expected shipping time. The solution to WLNR consists of two parts: the optimal location to introduce storage capacity, and the associated shipping policy. The shipping policy defines where to ship the cargo if multiple destinations with storage capacity (warehouses, sink) are reachable.

We show how WLNR is equivalent to a Markov Decision Process with Design (MDPD), where the design part comes from the decision where to place the warehouses. The MDPD has a state space that grows with network size, which makes the modes intractable for large networks. However, the solution reveals the optimal shipping policy; in particular, the cargo is shipped to the node from which one expects the smallest shipping time to the sink. We pre-define such a shipping policy. For fixed warehouse locations, the MDPD with pre-defined shipping policy behaves like a Markov chain. We use such a Markov chain to obtain a formula for the expected shipping time for one or two warehouses. The formula involve s-t-reliabilities, which are difficult to compute.

We provide methods to find the optimal warehouse locations for one and two warehouses. Estimating reliability by Monte Carlo simulation, which is very time consuming, lies at the core of the methods to solve WLNR. We show theoretical results that allow the elimination of suboptimal solutions from the solution space, resulting in a significant saving of computational effort. We create a case study from the DOD's humanitarian assistance transportation network with 95 nodes and 1096 arcs. There are more than 8000 possible ways to place two warehouses in that network. We identify up to 95% of these combinations as being suboptimal without computing the associated shipping times, which saves on simulation effort. We achieve further savings of simulation time by customizing the sampling method. We use a technique to sample the network and check the sample for a path from s to t in one step. This method only samples arcs as needed while searching for a path from s to t. In the case study, we sample less than 20% of all arcs when applying this technique. We also provide a theoretical bound on the best possible expected waiting time. The bound shows on the case study example that adding more than two warehouses improve results only fractionally.

WLNR only considers cargo and space-available with a fixed size of one unit. Future models could consider different cargo and unused space sizes, and potentially obtain other optimal solutions. Furthermore, WLNR is not focused on warehouse sizes, which is an important question to answer before building a warehouse. Including the decision about the storage capacity could be another direction to extend our work.

Appendix: Used Hardware and Software

To implemented all algorithms and models from this dissertation, we used the following hardware and software:

Intel®Core^{TM} i5-2400 CPU@3.10 GHz 3.10 GHz

8.00 GB RAM

OS Windows 7

GAMS build 23.9.3 with implemented CPLEX solver

Python 2.7.3 with Enthought Python Distribution (EDP) 7.3-2

THIS PAGE INTENTIONALLY LEFT BLANK

References

[1] T. D. O'Rourke, "Critical infrastructure, interdependencies, and resilience," *Bridge-Washington-National Academy of Engineering*, vol. 37, no. 1, pp. 22–29, 2007.

[2] L. Chen and E. Miller-Hooks, "Resilience: An indicator of recovery capability in intermodal freight transport," *Transportation Science*, vol. 46, no. 1, pp. 109–123, 2012.

[3] C. J. Colbourn, "Network resilience," *SIAM Journal on Algebraic Discrete Methods*, vol. 8, no. 3, pp. 404–409, 1987.

[4] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *The 5th International Workshop on Design of Reliable Communication Networks*, 2005, pp. 271–277.

[5] D. R. Fulkerson and G. C. Harding, "Maximizing the minimum source-sink path subject to a budget constraint," *Mathematical Programming*, vol. 13, no. 1, pp. 116–118, Dec. 1977.

[6] B. Golden, "A problem in network interdiction," *Naval Research Logistics Quarterly*, vol. 25, no. 4, pp. 711–713, Dec. 1978.

[7] E. Israeli and R. K. Wood, "Shortest-path network interdiction," *Networks*, vol. 40, no. 2, pp. 97–111, Sept. 2002.

[8] D. L. Alderson, G. G. Brown, M. W. Carlyle, and L. Anthony Cox, "Sometimes there is no "most-vital" arc: Assessing and improving the operational resilience of systems," *Military Operations Research*, vol. 18, no. 1, pp. 21–37, 2013.

[9] G. Brown, M. Carlyle, J. Salmeron, and K. Wood, "Defending critical infrastructure," *Interfaces*, vol. 36, no. 6, pp. 530–544, Nov. 2006.

[10] Quelle: dpa, "Castor-Transport, Nach langem Weg am Ziel," *Frankfurter Allgemeine Zeitung*, Nov. 28, 2011.

[11] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Transactions on Reliability*, vol. 35, no. 3, pp. 230–239, Aug. 1986.

[12] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, Aug. 1979.

[13] M. O. Ball and J. S. Provan, "Calculating bounds on reachability and connectedness in stochastic networks," *Networks*, vol. 13, no. 2, pp. 253–278, 1983.

[14] D. R. Shier and N. Liu, "Bounding the reliability of networks," *Journal of the Operational Research Society*, vol. 43, no. 5, pp. 539–548, May 1992.

[15] L. E. Miller, J. J. Kelleher, and L. Wong, "Assessment of network reliability calculation methods," J. S. Lee Associates, Rockville, MD, Rep. JC-2097-FF, Jan. 1993.

[16] M. G. Bell, "Mixed route strategies for the risk-averse shipment of hazardous materials," *Networks and Spatial Economics*, vol. 6, no. 3, pp. 253–265, 2006.

[17] Y. Dadkar, L. Nozick, and D. Jones, "Routing of hazardous material shipments under the threat of terrorist attack," *Security and Environmental Sustainability of Multimodal Transport*, pp. 89–110, 2010.

[18] U. Kanturska, J.-D. Schmöcker, A. Fonzone, and M. G. Bell, "Improving reliability through multi-path routing and link defence," *Game Theoretic Risk Analysis of Security Threats*, vol. 128, pp. 199–227, 2008.

[19] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 1993.

[20] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, July 1971.

[21] E. L. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, vol. 18, no. 7, pp. 401–405, Mar. 1972.

[22] W. Matthew Carlyle and R. Kevin Wood, "Near-shortest and k-shortest simple paths," *Networks*, vol. 46, no. 2, pp. 98–109, 2005.

[23] C.-L. Li, S. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, Jan. 1990.

[24] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, Dec. 1962.

[25] J. D. Camm, A. S. Raturi, and S. Tsubakitani, "Cutting big M down to size," *Interfaces*, vol. 20, no. 5, pp. 61–66, 1990.

[26] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization.* Belmont, MA: Athena Scientific, 1997.

[27] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[28] P. M. Murray-Tuite and X. Fei, "A methodology for assessing transportation network terrorism risk with attacker and defender interactions," *Computer-Aided Civil and Infrastructure Engineering*, vol. 25, no. 6, pp. 396–410, Feb. 2010.

[29] M. G. H. Bell, U. Kanturska, J.-D. Schmöcker, and A. Fonzone, "Attacker-defender models and road network vulnerability." *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, no. 1872, pp. 1893–1906, June 2008.

[30] D. P. Morton, F. Pan, and K. J. Saeger, "Models for nuclear smuggling interdiction," *IIE Transactions*, vol. 39, no. 1, pp. 3–14, Jan. 2007.

[31] L. V. Snyder, M. P. Scaparra, M. S. Daskin, and R. L. Church, "Planning for disruptions in supply chain networks," *Tutorials in Operations Research*, 2006.

[32] M. Dozier and N. B. Dimitrov, "Analysis of humanitarian assistance cargo transportation," Naval Postgraduate School, Monterey, CA, Tech. Rep. NPS-OR-11-007, Jan. 2012.

[33] W. J. Baumol and P. Wolfe, "A warehouse-location problem," *Operations Research*, vol. 6, no. 2, pp. 252–263, 1958.

[34] Z. Drezner and H. W. Hamacher, *Facility Location: Applications and Theory.* Heidelberg: Springer, 2002.

[35] S. H. Owen and M. S. Daskin, "Strategic facility location: A review," *European Journal of Operational Research*, vol. 111, no. 3, pp. 423–447, 1998.

[36] M. Fischetti, M. Monaci, and D. Salvagnin, "Three ideas for the quadratic assignment problem," *Operations Research*, vol. 60, no. 4, pp. 954–964, 2012.

[37] P. B. Mirchandani and A. R. Odoni, "Locations of medians on stochastic networks," *Transportation Science*, vol. 13, no. 2, pp. 85–97, 1979.

[38] P. B. Mirchandani, "Locational decisions on stochastic networks," *Geographical Analysis*, vol. 12, no. 2, pp. 172–183, 1980.

[39] J. R. Weaver and R. L. Church, "Computational procedures for location problems on stochastic networks," *Transportation Science*, vol. 17, no. 2, pp. 168–180, 1983.

[40] A. M. Law, *Simulation Modeling and Analysis*, 4th ed. New York, NY: McGraw-Hill, 2007.

[41] P. Bratley, B. L. Fox, and L. E. Schrage, *A Guide to Simulation.* New York, NY: Springer, 1986.

[42] G. S. Fishman, “A Monte Carlo sampling plan for estimating network reliability,” *Operations Research*, vol. 34, no. 4, pp. 581–594, July 1986.

[43] ——, “A Comparison of four Monte Carlo methods for estimating the probability of s-t connectedness,” *IEEE Transactions on Reliability*, vol. 35, no. 2, pp. 145–155, 1986.

[44] ——, “Estimating the s-t reliability function using importance and stratified sampling,” *Operations Research*, vol. 37, no. 3, pp. 462–473, May 1989.

[45] R. Van Slyke and H. Frank, “Network reliability analysis: Part I,” *Networks*, vol. 1, no. 3, pp. 279–290, 1971.

[46] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. Hoboken, NJ: Wiley, 2007.

[47] N. B. Dimitrov and D. P. Morton, “Combinatorial design of a stochastic Markov decision process,” *Operations Research and Cyber-Infrastructure*, pp. 167–193, 2009.

[48] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, 7th ed. Belmont, MA: Duxbury Press, 2008.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California